Adaptively Secure Attribute-Based Encryption from Witness Encryption

Brent Waters
UT Austin and NTT Research

Daniel Wichs
Northeastern and NTT Research

November 26, 2024

Abstract

Attribute-based encryption (ABE) enables fine-grained control over which ciphertexts various users can decrypt. A master authority can create secret keys sk_f with different functions (circuits) f for different users. Anybody can encrypt a message under some attribute x so that only recipients with a key sk_f for a function such that $f(x) = 1$ will be able to decrypt. There are a number of different approaches toward achieving *selectively secure* ABE, where the adversary has to decide on the challenge attribute x ahead of time before seeing any keys, including constructions via bilinear maps (for NC1 circuits), learning with errors, or witness encryption. However, when it comes *adaptively secure* ABE, the problem seems to be much more challenging and we only know of two potential approaches: via the "dual systems" methodology from bilinear maps, or via indistinguishability obfuscation. In this work, we give a new approach that constructs adaptively secure ABE from witness encryption (along with statistically sound NIZKs and one-way functions). While witness encryption is a strong assumption, it appears to be fundamentally weaker than indistinguishability obfuscation. Moreover, we have candidate constructions of witness encryption from some assumptions (e.g., evasive LWE) from which we do not know how to construct indistinguishability obfuscation, giving us adaptive ABE from these assumptions as a corollary of our work.

1 Introduction

Attribute-Based Encryption (ABE) [SW05] is an advanced form of encryption where the user's ability to decrypt ciphertexts is governed by a policy attached to their key. In such a system a ciphertext encrypting a message m is associated with a attribute string x. A secret key in turn will be issued by some authority which associates it with some predicate function f to generate sk_f. Decryption semantics dictate that sk_f will be able to decrypt a ciphertext associated with attribute x if $f(x) = 1$. A system is informally said to be secure if no attacker can distinguish between an encryption of message m_0 from m_1 under attribute x^* so long as it only obtains secret keys for functions $f_1, \ldots, f_q$ where $f_i(x^*) = 0$. Over the past two decades ABE has emerged as an important construct for both encrypted access control as well as at tool for building other cryptographic primitives (e.g., [PRV12, GKP+13]).

The first constructions of Attribute-Based Encryption [SW05, GPSW06] utilized groups with efficiently computable bilinear maps and supported functions that could be expressed as boolean formulas or circuits of logarithmic depth in the security parameter. Several years later construction based on lattices [GVW13, BGG+14] emerged that were provably secure from the learning with errors (LWE) [Reg05] assumption. Remarkably, these construction supported policies that could be expressed as any circuit of a priori bounded depth and thus in principle of any function of fixed runtime. Around the same time a third avenue for realizing ABE systems manifested when Garg, Gentry, Sahai and Waters [GGSW13] proposed the concept of witness encryption and showed how to build ABE from it. Witness is encryption is a powerful, yet general primitive where one encrypts a message m to a statement z and decryption is achievable for any decryptor which knows a witness w such that $R(z, w) = 1$ for some family of relations indexed by the security parameter.

Proving security is a central and involved part of building ABE systems. All three (bilinear map, LWE and witness encryption) paths for realizing Attribute-Based Encryption first established solutions in the selective model of security where an attacker declares an attribute string x^* *before* seeing either the public parameters of the system or receiving any private keys. This notion is meaningful; however, if fails to capture many "real life" attacks where an attacker might somehow influence the attribute string of a ciphertext in a way that depends on such information. While we can bridge the gap from selective to adaptive security using a complexity leveraging guessing strategy in conjunction with subexponential hardness assumptions, this is somewhat unsatisfactory both from the stronger assumption requirement and from an intellectual understanding standpoint.

Over the years achieving adaptive security has borne out to be quite challenging. Unlike Identity-Based Encryption (IBE) [Sha84] which admits a varied number of approaches [BF01, BB04, Wat05, Gen06, Wat09, DG17, Tsa19], ABE systems must maintain the "structure" and semantics of the attribute string which rule out many hashing techniques. Going further it was formally shown [LW14] that one cannot prove adaptive security using "partitioning" reductions which were integral to proving security for many IBE schemes.[1]

The first solutions [LOS+10] for adaptively secure Attribute-Based Encryption applied the dual system encryption methodology of Waters [Wat09] using bilinear groups. In a dual system encryption proof, the challenge ciphertext is first changed to a semi-functional form. Following this each secret key issued will be changed one at a time to a semi-functional form which is inherently incompatible with the challenge ciphertext, but still compatible with all other normally generated ciphertexts. Unfortunately, to this point it has proven difficult to find adaptations of these ideas to either the LWE or witness encryption avenues described above. (One exception is the work of [Tsa19] that gives an ABE system for a subset functionality which is more expressive than IBE string matching, but well short of ABE for boolean formulas or circuits.) From the learning with errors side, the algebraic analogs of bilinear map tools have not come fully to fruition. While witness encryption is a powerful primitive in some ways, it is arguably quite limited in others. In particular, it lacks the "hidden computation" aspect that is present in the more powerful concept of indistinguishability obfuscation. As such the only solutions for achieving adaptively secure ABE beyond bilinear maps have required indistinguishability obfuscation or functional encryption [Wat15, ABSV15] which precisely rely on such hidden computation properties.

Our Results: Adaptive ABE from Witness Encryption. In this work, we construct adaptively secure attribute-based encryption from witness encryption along with statistically sound NIZKs

[1] A weaker notion called semi-adaptive security [BV16, GKW16] is known to be significantly easier to achieve, but appears to still be far from fully adaptive security.

and one-way functions. At a high level, we do so by showing how to employ dual system encryption techniques using witness encryption.

This is both an important and technically challenging endeavor. While we already had adaptive ABE from indistinguishability obfuscation (iO) [Wat15, ABSV15] have recently seen iO proven from "well founded" assumptions [JLS21], witness encryption appears to be a fundamentally weaker primitive than iO. For example, we have black-box separations showing that witness encryption does not generically imply iO [GMM17]. Furthermore, witness encryption may admit solutions from a broader set of cryptographic assumptions. Two recent examples include the witness encryption built from variants of the evasive LWE assumption [Tsa22, VWW22] as well as a direction towards achieving witness encryption from pairing free groups [BIOW20]. Therefore, we get adaptively secure ABE from (e.g.,) evasive LWE as a corollary of our work. Overall, similarly to the recent work of [FWW23], we view the construction of advanced cryptosystems from *plain* witness encryption rather than iO as a well motivated and worthwhile endeavour.

Technically, witness encryption does not seem to support any form of hidden computation and thus appears to be incompatible with developing dual system encryption type proofs where we want to incrementally and undetectably change the form of the challenge ciphertext and private keys to make them mutually exclusive in a working decryption operation. We surmount this challenge by developing new tools and techniques for bringing in "outside" cryptography primitives to augment witness encryption to allow for such an argument.

1.1 Technical Overview

Selective ABE from WE. The prior work of [GGSW13] constructed selectively secure ABE from witness encryption. The main idea behind their solution is to set the master public/secret key to be a the verification/signing key for a special type of signature scheme. The secret keys sk_f are signature of the functions f, and an encryption under an attribute x is a witness encryption that there exists some signature for some function f such that $f(x) = 1$. In the proof of security, we can indistinguishably "constrain" the special signature scheme on the challenge attribute x^* so that there only exist valid signatures π for functions f for which $f(x^*) = 0$. Then the security of witness encryption ensures that the message is hidden. The signature itself is implemented using statistically binding commitments and statistically sound NIZKs. Unfortunately, this proof strategy inherently only achieves selective security since we need to know the challenge attribute x^* when creating the master public key of the ABE.

Overview of Our Approach. While our approach can also be seen relying on a special form of constrained signatures instantiated from commitments and NIZKs, the way we use these to achieve adaptive security is more sophisticated and is inspired by dual-system techniques [Wat09, LOS+10]. There are three main elements of our construction: (a) we introduce a new notion called a *functional tag system*, (b) we use a functional tag system to construct adaptive ABE from witness encryption (together with statistically binding commitments and statistically sound NIZKs), (c) we show how to construct a functional tag system from one-way functions. We now elaborate on each of these elements one by one.[2]

[2]In the main body, we reverse the order and present (c) before (b), but for the introduction we prefer this ordering.

Functional Tag System. A *functional tag system* allows us to generate "input tags" tag_x for inputs x and "function tags" tag_f for functions (i.e., circuits) f. There is a dummy (D) way to generate such tags $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(x), \mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(f)$ randomly and independently of each other. There is also a smart (S) way to generate these using some common secret key tsk with $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x), \mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$. There is an efficient predicate $\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x)$ that checks if some pair of function/input tag "trigger". Dummy pairs of tags trigger with only negligible probability. Smart pairs of tags generated using a common key tsk always trigger if $f(x) = 1$. A fully adaptive adversary who gets to see a single input tag tag_x for an input x and many function tags tag_{f_i} for functions f_i cannot tell the difference between seeing all dummy tags versus all smart tags generated using a common key tsk as long as $f_i(x) = 0$ for all i.

ABE from WE via a Functional Tag System. We set the master public/secret key of the ABE to be the verification/signing key for a special type of "constrained" signature scheme, described later on. Each function secret key $\mathsf{sk}_f = (f, \mathsf{tag}_f, \pi)$ consists of a randomly generated "dummy function tag" $\mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(f)$ along with a signature π of the pair (f, tag_f). To encrypt a message under an attribute x, we generate a "dummy input tag" $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(x)$ and send it along with a witness encryption of the message under the NP statement "there exists some pair (f, tag_f) that has a valid signature π such that $f(x) = 1$ and $\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 0$".[3]

In the proof of security, we first switch to using "smart function tags" $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$ in the secret keys sk_f and a "smart input tag" $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$ in the challenge ciphertext, all generated using a common key tsk. By the adaptive security of the functional tag system, this is indistinguishable. We then indistinguishably "constrain" the special signature scheme so that valid signatures π only exist for pairs (f, tag_f) where $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$. Finally, we argue that the NP statement used for the witness encryption is false, and therefore witness encryption security ensures that the encrypted message is hidden. This holds because whenever π is a valid signature of (f, tag_f) then it must be the case that $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$, and if $f(x) = 1$, then it must also be the case that $\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 1$.

The special constrained signature scheme is constructed from statistically binding commitments and statistically sound NIZKs as follows. The verification key consist of two commitments $\mathsf{com}_0, \mathsf{com}_1$ to 0, along with the CRS of the NIZK; the signing key is a decommitment of com_0. The signature π for a pair (f, tag_f) is a NIZK proof that "either com_0 is a commitment to 0 or com_1 is a commitment to tsk and there is some randomness r such that $\mathsf{tag}_f = \mathsf{SFunctionTag}(\mathsf{tsk}, f; r)$". The NIZK proof is generated using the decomitment of com_0 as the witness. To constrain the signature, we set com_0 to be a commitment to 1, com_1 to be a commitment to tsk and we generate the NIZKs using the the decomitment to com_1 and the randomness used to generate tag_f as the witness. The corresponding constrained verification key and signatures are indistinguishable.

Functional Tag System from One-Way Functions. Finally, we construct a functional tag system from one-way functions using "blind garbled circuits" [BLSV18]. In blind garbled circuits, for any distribution over input x and circuit C for which $C(x)$ is uniformly random, the corresponding

[3]We note that, in contrast to the selectively secure ABE schemes from LWE of [GVW13, BGG+14], our ABE is not succinct and the encryption run-time and ciphertext size scales with the circuit size of the supported functions f. Constructing even selectively secure succinct ABE from Witness Encryption is an intriguing open problem.

garbled input/circuit pair $\widetilde{x}, \widetilde{C}$ look like uniformly random bits. We rely on a slightly more complex version of blind garbled circuits where the adversary can see many different garbled circuits $\widetilde{C}_i$ but only one garbled input $\widetilde{x}$; furthermore we allow *semi-adaptive security* where the circuits and the input can be chosen adaptively, *but* the challenge circuit must be chosen after the input. The detailed definition is somewhat cumbersome and we defer it to the main body, but we show that the basic "point and permute" construction of garbled circuits from one-way functions achieves this notion similarly to [BLSV18].

To construct a functional tag system from blind garbled circuits, we set dummy input tags tag_x and dummy function tags tag_f to be uniformly random values of appropriate size. To determine if a input/function tag pair $(\mathsf{tag}_f, \mathsf{tag}_x)$ "triggers" we interpret $\mathsf{tag}_x = \widetilde{x}$ as a garbled input and $\mathsf{tag}_f = (\widetilde{C}, t)$ as a garbled circuit together with a target value t of length security parameter, and output 1 if the evaluation of the garbled circuit $\widetilde{C}$ on the garbled input $\widetilde{x}$ produces the target value t. In the dummy case, this only happens with negligible probability, ensuring "dummy correctness". A smart input tag for x consists of a correctly garbled input $\mathsf{tag}_x = \widetilde{x}$, and a smart function tag for f consists of $\mathsf{tag}_f = (\widetilde{C}, t)$ where t is a random target value and $\widetilde{C}$ is a garbling of the circuit C that evaluates $f(x)$ and if the output is 1 it outputs the target value t else it outputs a random independent value u. This ensures that a smart input/function tag pair $\mathsf{tag}_x, \mathsf{tag}_f$ does trigger when $f(x) = 1$.

For security, we intuitively want to rely on blind garbled circuits to ensure that we can replace dummy function tags with smart ones in the case where $f(x) = 0$, by relying on the fact that the circuit $C(x)$ outputs a random independent value u in this case. However, there is an issue with adaptivity. Blind garbled circuits only provide *semi-adaptive* security, where the challenge circuit C must be chosen after the input x, while functional tag systems require *fully adaptive* security where the challenge functions f can be chosen before or after the input x. We resolve this issue using techniques developed in the study of adaptively secure garbled circuits [HJO+16]. In particular, we encrypt the garbled circuit with a "somewhere equivocal PRF" whose key is part of the input tag. For any circuit C chosen before the input x, this allows us to give a fake ciphertext inside tag_f and only later equivocate the garbled circuit $\widetilde{C}$ inside the ciphertext after the input x is chosen, in affect allowing $\widetilde{C}$ to depend on x inside the security proof. Therefore, we can rely on semi-adaptive security of the blind garbled circuits to achieve fully adaptive security of the functional tag system.

2 Preliminaries

For any integer $n \geq 1$, define $[n] = \{1, \ldots, n\}$. A function $\nu : \mathbb{N} \to \mathbb{N}$ is said to be negligible, denoted $\nu(n) = \mathrm{negl}(n)$, if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\nu(n) < 1/p(n)$. We use the abbreviation PPT for probabilistic polynomial time. For a finite set S, we write $a \leftarrow S$ to mean a is sampled uniformly randomly from S. For a randomized algorithm A, we let $a \leftarrow A(\cdot)$ denote the process of running $A(\cdot)$ and assigning the outcome to a; when A is deterministic, we write $a := A(\cdot)$ instead. For a randomized algorithm A we use the notation $a := A(\cdot; r)$ to denote the process of running the randomized algorithm A with some fixed randomness r. We denote the security parameter by λ. For two distributions X, Y parameterized by λ we say that they are computationally indistinguishable, denoted by $X \approx_c Y$ if for every PPT distinguisher D we have $|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| = \mathrm{negl}(\lambda)$.

2.1 Attribute Based Encryption (ABE)

We define an ABE scheme with adaptive security.

Definition 2.1 (Attribute-Based Encryption (ABE).)**.** *An ABE scheme a function class* $\mathcal{F}_\lambda \subseteq \{f : \{0,1\}^{n(\lambda)} \to \{0,1\}\}$ *consists of PPT procedures* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*: Generates a master public key* mpk *and master secret key* msk.
- $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*: Generates a function key* sk_f *for a function* $f \in \mathcal{F}_\lambda$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, b)$*: Given an attribute* $x \in \{0,1\}^{n(\lambda)}$ *and a bit* $b \in \{0,1\}$ *outputs a ciphertext* ct.
- $b := \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$*: Decrypts* ct *usng* sk_f.

We require correctness and adaptive security defined as follows:

- ***Correctness:*** *There is some negligible function* μ *such that for all* $\lambda \in \mathbb{N}$ *all* $f \in \mathcal{F}_\lambda$ *all* $x \in \{0,1\}^{n(\lambda)}$ *such that* $f(x) = 1$ *all* $b \in \{0,1\}$ *we have:*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = b \;:\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ sk_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, b) \end{array}\right] \geq 1 - \mu(\lambda).$$

- ***Adaptive Security:*** *We define the game* $\mathsf{ABEGame}^b_{\mathcal{A}}(1^\lambda)$ *between a challenger and an stateful adversary* $\mathcal{A}(1^\lambda)$ *as follows:*
  - *The challenger chooses* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* mpk *to* $\mathcal{A}$.
  - Pre-challenge key queries: *The adversary can make arbitrarily many queries* $f_i \in \mathcal{F}_\lambda$ *and the challenger responds with* $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$.
  - Challenge ciphertext: *The adversary chooses an attribute* $x \in \{0,1\}^{n(\lambda)}$ *such that* $f_i(x) = 0$ *for all pre-challenge key queries* f_i*, and the challenger responds with the challenge ciphertext* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, b)$.
  - Post-challenge key queries: *The adversary can make arbitrarily many additional queries* $f_i \in \mathcal{F}_\lambda$ *such that* $f_i(x) = 0$ *and the challenger responds with* $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$.
  - *The adversary output a bit* b' *which is the output of the game.*

  We require that for all PPT $\mathcal{A}$ *we have*

$$\left|\Pr[\mathsf{ABEGame}^0_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{ABEGame}^1_{\mathcal{A}}(1^\lambda) = 1\right| \leq \mathrm{negl}(\lambda).$$

An ABE for circuits *allows us to instantiate an ABE scheme for the function class* $\mathcal{C}^{s,n}_\lambda$ *consisting of boolean circuits of size* $s(\lambda)$ *with* $n(\lambda)$*-bit input, for any polynomials* $s(\lambda), n(\lambda)$.

2.2 Commitments

We define statistically binding commitments in the Common Reference String (CRS) model.

Definition 2.2 (Statistically Binding Commitments)**.** *A commitment scheme consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Commit})$ *with the following syntax:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*: generates a common reference string* crs.
- $\mathsf{com} := \mathsf{Commit}_{\mathsf{crs}}(b; r)$*: generates a commitment* com *to a bit* $b \in \{0,1\}$ *using randomness* $r \in \{0,1\}^\lambda$.

We require hiding and statistical binding:

- ***Hiding:*** *We have* $(\mathsf{crs}, \mathsf{com}_0) \approx_c (\mathsf{crs}, \mathsf{com}_1)$ *where* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{com}_b \leftarrow \mathsf{Commit}_{\mathsf{crs}}(b)$.
- ***Statistical Binding:*** *We say that a* crs *is* binding *if there do not exist any* r_0, r_1 *such that* $\mathsf{Commit}_{\mathsf{crs}}(0; r_0) = \mathsf{Commit}_{\mathsf{crs}}(1; r_1)$. *We require that:* $\Pr[\mathsf{crs} \text{ is binding} \;:\; \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)] = 1 - \mathrm{negl}(\lambda)$.

We abuse notation and write $\mathsf{Commit}_{\mathsf{crs}}(x)$ *for a string* $x \in \{0,1\}^\ell$ *to denote the process of committing to each bit of* x *separately.*

Naor's commitment scheme [Nao91] gives statistically binding commitments assuming only one-way functions. In particular, it constructs commitments from a pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$, where $\mathsf{Setup}(1^\lambda)$ outputs a uniformly random $\mathsf{crs} \leftarrow \{0,1\}^{3\lambda}$ and $\mathsf{Commit}_{\mathsf{crs}}(b; r) = \mathsf{PRG}(r) \oplus (b \cdot \mathsf{crs})$. Hiding follows from PRG security and binding follows since

$$\Pr_{\mathsf{crs}}[\exists r_0, r_1 \;:\; \mathsf{PRG}(r_0) = \mathsf{PRG}(r_1) \oplus \mathsf{crs}] \le \sum_{r_0, r_1} \Pr[\mathsf{crs} = \mathsf{PRG}(r_0) \oplus \mathsf{PRG}(r_1)] \le 2^{2\lambda}/2^{3\lambda} \le 2^{-\lambda}.$$

Theorem 2.3 ([Nao91])**.** *Assuming one-way functions, there exist statistically binding commitments.*

2.3 NIZKs

We define statistically sound NIZKs in the CRS model with witness indistinguishability.

Definition 2.4 (Statistically Sound Non-Interactive Zero-Knowledge (NIZK))**.** *A NIZK proof system for an NP relation* $R_\lambda \subseteq \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}$ *with a corresponding NP language* $L_\lambda = \{x : \exists w \; (x, w) \in R_\lambda\}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*: generates a common reference string* crs.
- $\pi \leftarrow \mathsf{Prove}_{\mathsf{crs}}(x, w)$*: generates a proof* π *for the statement* x *using witness* w.
- $b = \mathsf{Verify}_{\mathsf{crs}}(x, \pi)$*: verifies the proof* π *for a given statement* x *and outputs a decision bit* 0 *(reject) or* 1 *(accept).*

We require the following properties:

- ***Completeness:*** *There exists a negligible function* μ *such that for all* $\lambda \in \mathbb{N}$*, all* $(x, w) \in R_\lambda$ *we have:*

$$\Pr\left[\mathsf{Verify}_{\mathsf{crs}}(x, \pi) = 1 \;:\; \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prove}_{\mathsf{crs}}(x, w)\right] \ge 1 - \mu(\lambda).$$

- ***Statistical Soundness:*** *We say that a* crs *is sound if for all* $x \notin L_\lambda$ *and all* π *we have* $\mathsf{Verify}_{\mathsf{crs}}(x, \pi) = 0$. *We require that* $\Pr[\mathsf{crs} \text{ is sound } : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)] = 1 - \mathrm{negl}(\lambda)$.
- ***Witness Indistinguishability:*** *For any ensemble* $x_\lambda, w_\lambda^0, w_\lambda^1$ *such that* $(x_\lambda, w_\lambda^b) \in R_\lambda$ *for* $b \in \{0, 1\}$ *we have* $(\mathsf{crs}, \pi_0) \approx_c (\mathsf{crs}, \pi_1)$ *where* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\pi_b \leftarrow \mathsf{Prove}_{\mathsf{crs}}(x_\lambda, w_\lambda^b)$ *for* $b \in \{0, 1\}$.

A NIZKs for NP *allows us to instantiate NIZK for any polynomial-time NP relation* R_λ. *We remark that the property of witness indistinguishability is weaker than an implied by the zero knowledge property typically associated with NIZKs.*

Theorem 2.5 ([FLS90, CHK03, GOS06, CCH+19, PS19])**.** *Statistically sound NIZKs in the CRS model exist assuming any one of: (1) hardness of factoring, or (2) the decisional linear assumption in bilinear groups, or (3) the learning with errors assumption.*

2.4 Witness Encryption

We define a witness encryption scheme.

Definition 2.6 (Witness Encryption)**.** *A witness encryption scheme for an NP relation* $R_\lambda \subseteq \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}$ *with a corresponding NP language* $L_\lambda = \{x \;:\; \exists w \;\; (x, w) \in R_\lambda\}$ *consists of PPT algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, b)$*: Encrypts a bit* $b \in \{0, 1\}$ *under the NP statement* $x \in \{0,1\}^{n(\lambda)}$.
- $b = \mathsf{Dec}(\mathsf{ct}, w)$*: Decrypts the ciphertexts using a witness* w.

We require the following properties:

- ***Correctness:*** *There exists a negligible function* μ *such that for all* $\lambda \in \mathbb{N}$*, all* $(x, w) \in R_\lambda$ *we have:*
$$\Pr\left[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, b), w) = b\right] \geq 1 - \mu(\lambda).$$
- ***Security :*** *For any ensemble* $\{x_\lambda\}_{\lambda \in \mathbb{N}}$ *such that* $x_\lambda \notin L_\lambda$ *for all* λ*, we have*
$$\mathsf{Enc}(1^\lambda, x_\lambda, 0) \approx_c \mathsf{Enc}(1^\lambda, x_\lambda, 1).$$

A WE for NP *allows us to instantiate WE for any polynomial-time NP relation* R_λ.

We abuse notation and write $\mathsf{Enc}(1^\lambda, x, m)$ for a long message $m \in \{0,1\}^\ell$ to denote the process of encrypting the message bit-wise $\mathsf{Enc}(1^\lambda, x, m_1), \ldots, \mathsf{Enc}(1^\lambda, x, m_\ell)$.

2.5 Somewhere equivocal PRF

We define the notion of a somewhere equivocal PRF (SEPRF) from [HJO+16, Definition 7] (also refereed to as a 1-SEPRF there). Intuitively, an SEPRF consists of a pseudorandom function $y = \mathsf{PRF}(\mathsf{key}, x)$ that maps inputs x to outputs y using a secret key. For any input x^*, there is a way to generate an equivocal key eqKey that leaves the output of the PRF unspecified at x^*, but allows us to evaluate it at all input $x \neq x^*$ by computing $\mathsf{PRF}(\mathsf{eqKey}, x)$. Later for any output y^* we can fix the output of the PRF at x^* to y^* by generating a key key such that $\mathsf{PRF}(\mathsf{key}, x^*) = y^*$, while ensuring $\mathsf{PRF}(\mathsf{key}, x) = \mathsf{PRF}(\mathsf{eqKey}, x)$ for all $x \neq x^*$. Moreover, for any x^*, one cannot distinguish between an honestly generated key versus first generating eqKey that is equivocal at x^* and later fixing the output of the PRF at x^* to a uniformly random y^* by generating the corresponding key.[4]

Definition 2.7 (SEPRF). *An SEPRF with input length $n(\lambda)$ and output length $m(\lambda)$ consists of the PPT algorithms* $(\mathsf{KeyGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *with the following syntax:*

- $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$*: generates a PRF key.*
- $y = \mathsf{PRF}(\mathsf{key}, x)$*: A deterministic algorithm that takes as input $x \in \{0,1\}^{n(\lambda)}$ and outputs $y \in \{0,1\}^{m(\lambda)}$.*
- $\mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, x^*)$*: Given $x^* \in \{0,1\}^{n(\lambda)}$ outputs a key* eqKey *that is equivocal on x^*.*
- $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, y^*)$*: Given an output $y^* \in \{0,1\}^{m(\lambda)}$ creates an equivocated key* key.

We require two properties:

- ***Correctness:*** *For all $x^* \in \{0,1\}^{n(\lambda)}$, $y^* \in \{0,1\}^{m(\lambda)}$ we have*

$$\Pr\left[\begin{array}{ll} & \mathsf{PRF}(\mathsf{key}, x^*) = y^* \\ \wedge & \forall x \neq x^* \;:\; \mathsf{PRF}(\mathsf{key}, x) = \mathsf{PRF}(\mathsf{eqKey}, x) \end{array} : \begin{array}{l} \mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, x^*) \\ \mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, y^*) \end{array}\right] = 1.$$

- ***Security:*** *We define the game $\mathsf{SEPRFGame}^b_{\mathcal{A}}(1^\lambda)$ between a challenger and an stateful adversary $\mathcal{A}(1^\lambda)$ as follows:*
  - *The adversary chooses $x^* \in \{0,1\}^{n(\lambda)}$.*
  - *If $b = 0$ the challenger chooses $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives* key *to the adversary.*
    If $b = 1$ the challenger chooses $\mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, x^*)$, $y^* \leftarrow \{0,1\}^{m(\lambda)}$, $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, y^*)$ and gives key *to the adversary.*
  - *The adversary outputs a bit b' which is the output of the game.*

  We require that for all PPT $\mathcal{A}$ we have

$$\left|\Pr[\mathsf{SEPRFGame}^0_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{SEPRFGame}^1_{\mathcal{A}}(1^\lambda) = 1\right| \leq \mathrm{negl}(\lambda).$$

Theorem 2.8 ([HJO+16]). *Assuming the existence of one-way functions, for any polynomials $n = n(\lambda), m = m(\lambda)$ there exists an SEPRF with input length n and output length m.*

[4] Our definition below is slightly syntactically simplified from the one in [HJO+16] since we have Sim_1 output a single value eqKey while the one in [HJO+16] outputs a pair $\mathsf{key}', \mathsf{state}$ where the former is used to evaluate the PRF and the latter is used by Sim_2. However, we can always set $\mathsf{eqKey} = (\mathsf{key}', \mathsf{state})$ and have the PRF evaluation ignore the second component to derive a scheme matching our syntax. Note that there is no requirement that eqKey looks like key.

3 Functional Tag System

Our paper consists of three main components: (1) introducing a new notion of *functional tag systems* in this section, (2) constructing a functional tag system from one-way functions via garbled circuits in Section 4, and (3) constructing adaptively secure ABE from WE via a functional tag system in Section 5.

A *functional tag system* allows us to generate "input tags" tag_x for inputs x and "function tags" tag_f for functions f. There is a dummy (D) way to generate these randomly/independently and a smart (S) way to generate these in a coordinated way using some common secret key tsk. There is an efficient procedure that checks if some combinations of $(\mathsf{tag}_f, \mathsf{tag}_x)$ "trigger". Dummy ones trigger with negligible probability. Smart ones always trigger if $f(x) = 1$. A fully adaptive adversary who gets to see a single input tag for an input x and many function tags for functions f_i cannot tell the difference between dummy and smart as long as $f_i(x) = 0$ for all i.

Definition 3.1 (Functional Tag System)**.** *A functional tag system for a function class $\mathcal{F}_\lambda \subseteq \{f : \{0,1\}^{n(\lambda)} \to \{0,1\}\}$ consists of PPT procedures*

$$(\mathsf{DInputTag}, \mathsf{DFunctionTag}, \mathsf{SGen}, \mathsf{SInputTag}, \mathsf{SFunctionTag}, \mathsf{Trigger})$$

with the following syntax:

- $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(1^\lambda, x)$ *takes as input $x \in \{0,1\}^{n(\lambda)}$ and generates a "dummy input tag".*
- $\mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(1^\lambda, f)$ *takes as input $f \in \mathcal{F}_\lambda$ and generates a "dummy function tag".*
- $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$ *generates a tag key* tsk.
- $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$ *takes as input $x \in \{0,1\}^{n(\lambda)}$ and generates a "smart input tag".*
- $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$ *takes as input $f \in \mathcal{F}_\lambda$ and generates a "smart function tag".*
- $b = \mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x)$ *a deterministic procedure that outputs 0 (not triggered) or 1 (triggered).*

The scheme has the following properties:

1. ***Dummy Correctness:*** *There exists some negligible μ such that for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda, x \in \{0,1\}^{n(\lambda)}$:*

$$\Pr\left[\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 1 \ : \ \begin{array}{l} \mathsf{tag}_x \leftarrow \mathsf{DInputTag}(x) \\ \mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(f) \end{array}\right] \leq \mu(\lambda).$$

2. ***Smart Correctness:*** *For all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda, x \in \{0,1\}^{n(\lambda)}$ such that $f(x) = 1$:*

$$\Pr\left[\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 1 \ : \ \begin{array}{l} \mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda) \\ \mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x) \\ \mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f) \end{array}\right] = 1.$$

3. ***Security:*** *We define the game $\mathsf{FunTagGame}^b_{\mathcal{A}}(1^\lambda)$ between a challenger with a bit b and an stateful adversary $\mathcal{A}(1^\lambda)$ as follows:*

   - *If $b = 1$, the challenger samples a random* $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$.

- Pre-challenge function tag queries: *The adversary can make arbitrarily many queries* $f_i \in \mathcal{F}_\lambda$*. If* $b = 0$ *the challenger responds with* $\mathsf{tag}_{f_i} \leftarrow \mathsf{DFunctionTag}(1^\lambda, f_i)$ *and if* $b = 1$ *the challenger responds with* $\mathsf{tag}_{f_i} \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f_i)$.
- Challenge input tag: *The adversary chooses an input* $x \in \{0,1\}^{n(\lambda)}$ *such that* $f_i(x) = 0$ *for all prior function tag queries* f_i*. If* $b = 0$ *the challenger responds with* $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(1^\lambda, x)$ *and if* $b = 1$ *the challenger responds with* $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$.
- Post-challenge function tag queries: *The adversary can make arbitrarily many additional queries* $f_i \in \mathcal{F}_\lambda$ *such that* $f_i(x) = 0$*. If* $b = 0$ *the challenger responds with* $\mathsf{tag}_{f_i} \leftarrow \mathsf{DFunctionTag}(1^\lambda, f_i)$ *and if* $b = 1$ *the challenger responds with* $\mathsf{tag}_{f_i} \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f_i)$.
- *The adversary output a bit* b' *which is the output of the game.*

We require that for all PPT $\mathcal{A}$ *we have*

$$\left|\Pr[\mathsf{FunTagGame}^0_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{FunTagGame}^1_{\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda).$$

A functional tag system for circuits *allows us to instantiate a functional tag system for the class* $\mathcal{C}^{s,n}_\lambda$ *consisting of boolean circuits of size* $s(\lambda)$ *with* $n(\lambda)$*-bit input, for any polynomials* $s(\lambda), n(\lambda)$.

4 A Functional Tag System from One-Way Functions

We construct a functional tag system for circuits from one-way functions. Our main tool is a special form of blind garbled circuits. We first define and construct this form of blind garbled circuits and then proceed to use them to construct a functional tag system.

4.1 Blind Garbled Circuits

We rely on blind garbled circuits, originally defined in [BLSV18]. In a blind garbled circuits, if one gets a garbled input together with a garbled circuit that outputs a uniformly random value on that input, the pair looks like completely random bits. We rely on a variant of blind garbled circuit security that we call *semi-adaptive blind garbled circuits*, defined formally below. Informally, we consider a game where the adversary can get arbitrarily many garbled circuits and a single garbled input $\widetilde{x}$ of a value x, all chosen adaptively. In addition the adversary chooses a challenge circuit C *after* it gets the garbled input $\widetilde{x}$ and should not be able to distinguish between a real garbling $\widetilde{C}$ of the challenge circuit C versus a simulated one. Note that the input x cannot depend on the garbled circuit $\widetilde{C}$, which avoids the main difficulty in adaptively secure garbled circuits. The simulator needs to simulate the garbled circuit $\widetilde{C}$ given $x, C(x)$, but without knowing the circuit C. For blindness, we require that, for a uniformly random output $C(x)$, the corresponding simulated garbled circuit $\widetilde{C}$ is uniformly random. While the definition is incomparable to the one in [BLSV18], we show that the same "point-and-permute" construction used in [BLSV18] satisfies our definition as well.

Definition 4.1 (Semi-adaptive Blind Garbled Circuit)**.** *Let* $\mathcal{C}^{s,n,m}_\lambda$ *be a class of circuits of size* $s = s(\lambda)$ *with* $n = n(\lambda)$*-bit input and* $m = m(\lambda)$*-bit output. A* semi-adaptive blind garbled circuit *scheme for* $\mathcal{C}^{s,n,m}_\lambda$ *consist of PPT algorithms:* $(\mathsf{GarbleGen}, \mathsf{GInput}, \mathsf{GCircuit}, \mathsf{SimCircuit}, \mathsf{Eval})$ *and garbled circuit size parameter* $\ell = \ell(\lambda)$ *with the following syntax.*

- $\mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda)$*: generates a garbling secret key* sk.
- $\widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x)$*: garbles an input* $x \in \{0,1\}^n$.
- $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C)$*: garbles a circuit* $C \in \mathcal{C}_\lambda^{s,n,m}$ *with* $\widetilde{C} \in \{0,1\}^\ell$.
- $y := \mathsf{Eval}(\widetilde{C}, \widetilde{x})$*: a deterministic algorithm that evaluates the garbled circuit on the garbled input and yields output* $y \in \{0,1\}^m$.
- $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, y)$*: produces a simulated circuit* $\widetilde{C} \in \{0,1\}^\ell$ *for a given output* $y = C(x)$ *without knowing* C.

We require the following properties:

Correctness: *For all* $\lambda, C \in \mathcal{C}_\lambda^{s,n,m}$ $x \in \{0,1\}^n$ *we have*

$$\Pr[\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = C(x) \ : \ \mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda), \widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x), \widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C)] = 1.$$

Semi-Adaptive Simulation Security: *We define the game* $\mathsf{GCGame}^b_{\mathcal{A}}(1^\lambda)$ *between a challenger with a bit b and an stateful adversary* $\mathcal{A}(1^\lambda)$ *as follows:*

- *Challenger picks* $\mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda)$.
- *Adversary* $\mathcal{A}^{\mathsf{GCircuit}(\mathsf{sk},\cdot)}$ *picks* $x \in \{0,1\}^n$ *and gets back* $\widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x)$.
- *Adversary* $\mathcal{A}^{\mathsf{GCircuit}(\mathsf{sk},\cdot)}$ *picks a boolean circuit* $C \in \mathcal{C}_\lambda^{n,m}$. *The adversary gets back either* $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C)$ *if* $b = 0$ *or* $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, C(x))$ *if* $b = 1$.
- *Adversary* $\mathcal{A}^{\mathsf{GCircuit}(\mathsf{sk},\cdot)}$ *outputs a bit* b' *which is the output of the game.*

We require that for all PPT $\mathcal{A}$ *we have*

$$\left|\Pr[\mathsf{GCGame}^0_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{GCGame}^1_{\mathcal{A}}(1^\lambda) = 1]\right| \leq \operatorname{negl}(\lambda).$$

Blindness: *For every fixed choice of* sk *in the support of* $\mathsf{GarbleGen}(1^\lambda)$ *every* $x \in \{0,1\}^n$ *we have:*

$$\mathsf{SimCircuit}(\mathsf{sk}, x, U_m) \equiv U_\ell,$$

where U_i *denotes the uniform distribution over* $\{0,1\}^i$ *and "*$\equiv$*" denotes distributional equivalence.*

Construction. Let $s(\lambda), n(\lambda), m(\lambda)$ be arbitrary polynomials. We assume that circuits in $\mathcal{C}_\lambda^{s,n,m}$ *have some fixed topology*. In particular, each circuit $C \in \mathcal{C}_\lambda^{s,n,m}$ consists of s gates and $s + n + m$ wires, with n input wires denoted $\mathsf{in}_1, \dots \mathsf{in}_n$, m output wires denoted $\mathsf{out}_1, \dots, \mathsf{out}_m$ and s internal wires. Each gate $g \in [s]$ gets 2 input wires and 1 output wire; we allow arbitrary fan-out since each output wire can be an input to arbitrarily many other gates. Each gate g computes some function $f_g \ : \ \{0,1\}^2 \to \{0,1\}$. The gates are connected via some fixed topology that is the same for all circuits in the class: that is, any gate $g \in [s]$ has some fixed input writes $w_{g,1}, w_{g,2}$ and output wire $w_{g,w}$ for all $C \in \mathcal{C}_\lambda^{s,n,m}$. The only distinction between different circuits $C \in \mathcal{C}_\lambda^{s,n,m}$ are the functions f_g computed by each gate. Note that we can convert general circuits into ones having a fixed topology with only a polylogarithmic blowup in circuit size via *universal circuits*, and therefore the above assumption is without loss of generality. Let $\mathsf{PRF} \ : \ \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{\lambda+1}$ be a pseudorandom function. The "point-and-permute" construction of blind garbled circuits for the class $\mathcal{C}_\lambda^{s,n,m}$ works as follows:

- $\mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda)$: For each of the n input wires in_i, sample PRF keys $s_{\mathsf{in}_i,b} \leftarrow \{0,1\}^\lambda$ and random bits $\alpha_{\mathsf{in}_i} \leftarrow \{0,1\}$ for $i \in [n], b \in \{0,1\}$. Let $\mathsf{sk} = (s_{\mathsf{in}_i,b}, \alpha_{\mathsf{in}_i})_{i\in[n],b\in\{0,1\}}$.

- $\widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x)$: For $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ output $\widetilde{x} = (s_{\mathsf{in}_i,x_i}, \alpha_{\mathsf{in}_i} \oplus x_i)_{i\in[n]}$.

- $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C)$: Sample a random circuit nonce $c \leftarrow \{0,1\}^\lambda$. For each wire w that is not an input wire sample fresh PRF keys $s_{w,b} \leftarrow \{0,1\}^\lambda$ for $b \in \{0,1\}$ along with a random bit $\alpha_w \leftarrow \{0,1\}$. The corresponding values $s_{\mathsf{in}_i,b}, \alpha_{\mathsf{in}_i}$ for the input wires in_i are contained in sk. For each gate g, let $f_g : \{0,1\}^2 \to \{0,1\}$ be the Boolean function computed by the gate. For every gate $g \in [s]$ with input wires w_1, w_2 and output wire w_3, and for all $\beta_1, \beta_2 \in \{0,1\}$, set $\beta_3 := f_g(\alpha_{w_1} \oplus \beta_1, \alpha_{w_2} \oplus \beta_2) \oplus \alpha_{w_3}$, and compute the table entry:

  $$T_g^{\beta_1,\beta_2} := (s_{w_3,\alpha_{w_3}\oplus\beta_3} \parallel \beta_3) \oplus \mathsf{PRF}(s_{w_1,\alpha_{w_1}\oplus\beta_1}, c \parallel g \parallel \beta_1 \parallel \beta_2) \oplus \mathsf{PRF}(s_{w_2,\alpha_{w_2}\oplus\beta_2}, c \parallel g \parallel \beta_1 \parallel \beta_2). \tag{1}$$

  Define the table for gate g as $T_g := (T_g^{\beta_1,\beta_2})_{\beta_1,\beta_2\in\{0,1\}}$. Then, output the garbled circuit consisting of:

  $$\widetilde{C} = \big(c \ , \ (T_g)_{g\in[s]} \ , \ (\alpha_{\mathsf{out}_j})_{j\in[m]} \ \big),$$

  with $\widetilde{C} \in \{0,1\}^\ell$ for $\ell = \lambda + 4(\lambda+1)s + m$.

- $y := \mathsf{Eval}(\widetilde{C}, \widetilde{x})$: Parse $\widetilde{x} = (s_{\mathsf{in}_i}, \beta_{\mathsf{in}_i})_{i\in[n]}$. For every gate $g \in [s]$ in topological order, let w_1, w_2 be its input wires and let w_3 be its output wire. Then, given $(s_{w_1} \parallel \beta_{w_1}), (s_{w_2} \parallel \beta_{w_2})$, compute

  $$(s_{w_3} \parallel \beta_{w_3}) = T_g^{\beta_{w_1},\beta_{w_2}} \oplus \mathsf{PRF}(s_{w_1}, c \parallel g \parallel \beta_{w_1} \parallel \beta_{w_2}) \oplus \mathsf{PRF}(s_{w_2}, c \parallel g \parallel \beta_{w_1} \parallel \beta_{w_2}).$$

  Finally, upon obtaining β_{out_j}, set $y_j := \beta_{\mathsf{out}_j} \oplus \alpha_{\mathsf{out}_j}$ for $j \in [m]$ and output $y := (y_1, \ldots, y_m) \in \{0,1\}^m$.

- $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, y)$: Sample a random circuit nonce $c \leftarrow \{0,1\}^\lambda$. For each wire w that is not an input wire sample a fresh PRF key $s_w \leftarrow \{0,1\}^\lambda$ along with a random bit $\beta_w \leftarrow \{0,1\}$. For each input wire in_i, set $s_{\mathsf{in}_i} := s_{\mathsf{in}_i,x_i}$ $\beta_{\mathsf{in}_i} := x_i \oplus \alpha_{\mathsf{in}_i}$ using the values from sk. For each gate g with input wires w_1, w_2 and output wire w_3, compute

  $$T_g^{\beta_{w_1},\beta_{w_2}} := (s_{w_3} \parallel \beta_{w_3}) \oplus \mathsf{PRF}(s_{w_1}, c \parallel g \parallel \beta_{w_1} \parallel \beta_{w_2}) \oplus \mathsf{PRF}(s_{w_2}, c \parallel g \parallel \beta_{w_1} \parallel \beta_{w_2}), \tag{2}$$

  and choose $T_g^{\beta_0,\beta_1} \leftarrow \{0,1\}^{\lambda+1}$ uniformly at random for all $(\beta_0, \beta_1) \neq (\beta_{w_1}, \beta_{w_2})$. Define the table for gate g as $T_g := (T_g^{\beta_1,\beta_2})_{\beta_1,\beta_2\in\{0,1\}}$. Output

  $$\widetilde{C} = \big(c \ , \ (T_g)_{g\in[s]} \ , \ (\beta_{\mathsf{out}_j} \oplus y_j)_{j\in[m]} \ \big).$$

Theorem 4.2. *Assuming one-way functions, there exist semi-adaptive blind garbled circuits for the class $\mathcal{C}_\lambda^{s,n,m}$ for any polynomials s, n, m.*

Proof. We show that the "point-and-permute" construction from pseudorandom functions described above is a semi-adaptive garbled circuit. The theorem then follows using the fact that pseudorandom functions can be constructed from one-way functions.

Perfect Correctness. For the perfect correctness of the construction, note that during evaluation it holds that each computed value $(s_w, \beta_w) = (s_{w,\mathsf{val}(w)}, \mathsf{val}(w) \oplus \alpha_w)$, where $\mathsf{val}(w)$ is the value on the wire w during the computation $C(x)$, and $s_{w,b}, \alpha_w$ are the values chosen during garbling. This is true for the input wires, and is easily seen to be true for all subsequent wires by induction. Therefore it holds that for the output wires $\beta_{\mathsf{out}_j} = y_j \oplus \alpha_{\mathsf{out}_j}$ and therefore evaluation computes the correct outputs y_j.

Semi-Adaptive Simulation Security. To prove semi-adaptive simulation security, we do a sequence of hybrids where we change the challenge garbled circuit $\widetilde{C}$ from real to simulated. Firstly, we define the games $\widehat{\mathsf{GCGame}}^b$ identically to GCGame^b, except that the game outputs 0 if the circuit nonce c used in the challenge garbled circuit $\widetilde{C}$ is ever used in any other garbled circuit created by the $\mathsf{GCircuit}(\mathsf{sk}, \cdot)$ oracle. For $b \in \{0,1\}$, the games GCGame^b and $\widehat{\mathsf{GCGame}}^b$ are statisitically indistinguishable. Therefore it suffices to show that $\widehat{\mathsf{GCGame}}^0$ and $\widehat{\mathsf{GCGame}}^1$ are computationally indistinguishable.

To do so, we iterate over all gates $g \in [s]$ in topological order starting with the input layer. For $i \in [s+1]$, define hybrids Game_i where the challenge garbled circuit

$$\widetilde{C} = \left(c \ , \ (T_g)_{g \in [s]} \ , \ (\alpha_{\mathsf{out}_j})_{j \in [m]} \ \right)$$

is sampled as follows. We sample the values $c, s_{w,b}, \alpha_w$ as specified by $\mathsf{GCircuit}$. Define $s_w := s_{w,\mathsf{val}(w)}, \beta_w := \alpha_w \oplus \mathsf{val}(w)$ where $\mathsf{val}(w)$ is the value on the wire w during the computation $C(x)$, which is well defined since the input x is chosen before the challenge circuit $\widetilde{C}$ is created. For the gates $g \geq i$ the tables $T_g := (T_g^{\beta_1,\beta_2})_{\beta_1,\beta_2 \in \{0,1\}}$ are created as in $\mathsf{GCircuit}$ following equation 1. For gates $g < i$, the tables $T_g := (T_g^{\beta_1,\beta_2})_{\beta_1,\beta_2 \in \{0,1\}}$ are instead created as in $\mathsf{SimCircuit}$; namely if the gate g has input wires w_1, w_2 and output wire w_3, then the table entry $T_g^{\beta_{w_1},\beta_{w_2}}$ is created as in equation 2 and the other entries are sampled randomly with $T_g^{\beta_0,\beta_1} \leftarrow \{0,1\}^{\lambda+1}$ for all $(\beta_0, \beta_1) \neq (\beta_{w_1}, \beta_{w_2})$. It is easy to see that Game_1 is identical to $\widehat{\mathsf{GCGame}}^0$. Furthermore, for all $g \in [s]$, Game_g is computationally indistinguishable from Game_{g+1}. The only difference between the games is how the entries $T_g^{\beta_0,\beta_1}$ for $(\beta_0, \beta_1) \neq (\beta_{w_1}, \beta_{w_2})$ are sampled. However, it is easy to show that the games are indistinguishable by PRF security. In particular, for these entries, at least one of the two PRF outputs in equation 1 involves a PRF key $s_{w,b}$ that is not used in the game in any other way beyond black-box PRF evaluation $\mathsf{PRF}(s_{w,b}, \cdot)$ and the input $c \parallel g \parallel \beta_1 \parallel \beta_2$ on which the PRF is evaluated is not used anywhere else. Therefore, we can replace this PRF output by uniform. Lastly, we observe that Game_{s+1} is identical to $\widehat{\mathsf{GCGame}}^1$. This simply follows since, for each non-input wire, the values $s_w := s_{w,\mathsf{val}(w)}, \beta_w := \alpha_w \oplus \mathsf{val}(w)$ are uniformly random over the choice of $s_{w,0}, s_{w,1}, \alpha_w$ and for the output wires we have $\alpha_{\mathsf{out}_j} = \beta_{\mathsf{out}_j} \oplus \mathsf{val}(\mathsf{out}_j) = \beta_{\mathsf{out}_j} \oplus y_j$.

Blindness. Finally, to show blindness, we need to show that the distribution of the simulated garbled circuit

$$\widetilde{C} = \left(c \ , \ (T_g)_{g \in [s]} \ , \ (\beta_{\mathsf{out}_j} \oplus y_j)_{j \in [m]} \ \right) \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, U_m)$$

satisfies $\widetilde{C} \equiv \{0,1\}^\ell$ for $\ell = \lambda + 4(\lambda+1)s + m$. First, $\widetilde{C} \equiv \left(c \ , \ (T_g)_{g \in [s]} \ , \ U_m \ \right)$ since the $y \leftarrow U_m$ is uniformly random and independent of $c, (T_g)_{g \in [s]}$ or $\{\beta_w\}$. Second, we proceed in reverse topological order starting at the output layer and show that each table T_g is uniformly random over $\{0,1\}^{4(\lambda+1)}$ even given c and all the tables T_i for $i < g$. This follows from equation 2 and the fact that $(s_{w_3} \parallel \beta_{w_3})$ is uniformly random and is not used in the construction of tables T_i for $i < g$. Therefore $\widetilde{C} \equiv \big(c \ , \ (T_g)_{g \in [s]} \ , \ U_m \ \big) \equiv (c, U_{4(\lambda+1)s}, U_m) \equiv U_\ell$.

□

4.2 Functional Tag System from Blind Garbled Circuits

We construct a functional tag system (Def. 3.1) from one-way functions using blind garbled circuits (Def. 4.1) and a somewhere equivocal PRF (Def. 2.7). As a starting point of our construction, we set dummy input tags to be garbled inputs $\mathsf{tag}_x = \widetilde{x}$ using a fresh garbling key sk, and dummy function tags $\mathsf{tag}_f = (\widetilde{C}, t)$ consist of a uniformly random garbled circuit $\widetilde{C} \leftarrow \{0,1\}^\ell$ along with some target value t. An input/function tag "triggers" if evaluating the garbled circuit $\widetilde{C}$ on the garbled input $\widetilde{x}$ produces the target value t. In the dummy case, this only happens with negligible probability, ensuring "dummy correctness". A smart input tag is a garbled inputs $\mathsf{tag}_x = \widetilde{x}$ using a garbling key sk contained in the tag key $\mathsf{tsk} = \mathsf{sk}$, and a smart function tag $\mathsf{tag}_f = (\widetilde{C}, t)$ consists of a random target value t along with a correctly garbled circuit $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{tsk}, C)$ of the circuit C that evaluates $f(x)$ and if the output is 1 it outputs the target value t else it outputs a random independent value u. This ensures that a smart input/function tag pair $\mathsf{tag}_x, \mathsf{tag}_f$ does trigger when $f(x) = 1$. For security, we intuitively want to rely on blind garbled circuits to ensure that we can replace dummy function tags with smart ones in the case where $f(x) = 0$, by relying on the fact that the circuit $C(x)$ outputs a random independent value u in this case. However, there is an issue with adaptivity. Blind garbled circuits only provide *semi-adaptive* security, where the challenge circuit C must be chosen after the input x, while functional tag systems require *fully adaptive* security where the challenge functions f can be chosen before or after the input x. We resolve this issue by encrypting the garbled circuit with a somewhere equivocal PRF whose key is part of the input tag. For any circuit C chosen before the input x, this allows us to give a fake ciphertext inside tag_f and only later equivocate the garbled circuit $\widetilde{C}$ inside the ciphertext after the input x is chosen, in affect allowing $\widetilde{C}$ to depend on x inside the security proof. Therefore, we can rely on semi-adaptive security of the blind garbled circuits to achieve fully adaptive security of the functional tag system.

Construction. Let $n = n(\lambda), s = s(\lambda)$ be any polynomials. We construct a functional tag system for the class $\mathcal{F}_\lambda = \mathcal{C}_\lambda^{s,n}$ consisting of circuits of size s with n-bit input and 1-bit output. Let $(\mathsf{GarbleGen}, \mathsf{GInput}, \mathsf{GCircuit}, \mathsf{Eval})$ be a semi-adaptive blind garbled circuit for the class $\mathcal{C}_\lambda^{s',n,m=\mathrm{sec}}$, where $s' = s + O(\lambda)$ will be defined later, and let $\ell = \ell(\lambda)$ be the corresponding garbled circuit size. Let $(\mathsf{KeyGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ be a somewhere equivocal PRF with input length λ and output length ℓ. We construct a functional tag system $(\mathsf{DInputTag}, \mathsf{DFunctionTag}, \mathsf{SGen}, \mathsf{SInputTag}, \mathsf{SFunctionTag}, \mathsf{Trigger})$ defined as follows:

- $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(x)$: Choose $\mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda)$, $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x)$. Output $\mathsf{tag}_x = (\mathsf{key}, \widetilde{x})$.
- $\mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(f)$: Output $\mathsf{tag}_f = (t_0, t_1, t_2) \leftarrow \{0,1\}^\lambda \times \{0,1\}^\ell \times \{0,1\}^\lambda$.
- $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$: Choose $\mathsf{sk} \leftarrow \mathsf{GarbleGen}(1^\lambda)$, $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and set $\mathsf{tsk} = (\mathsf{sk}, \mathsf{key})$.

- $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$: Choose $\widetilde{x} \leftarrow \mathsf{GInput}(\mathsf{sk}, x)$. Output $\mathsf{tag}_x = (\mathsf{key}, \widetilde{x})$.
- $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$: Choose $t_0, t_2, u \leftarrow \{0,1\}^\lambda$ and let C^*_{f,u,t_2} be the circuit that on input x outputs u if $f(x) = 0$ and outputs t_2 if $f(x) = 1$. We define the parameter $s' = s + O(\lambda)$ be the size of C^*_{f,u,t_2} for $f \in \mathcal{C}^{s,n}$. Let $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f,u,t_2})$ and set $t_1 = \mathsf{PRF}(\mathsf{key}, t_0) \oplus \widetilde{C}$. The output is $\mathsf{tag}_f = (t_0, t_1, t_2)$.
- $\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x)$: Parse $\mathsf{tag}_x = (\mathsf{key}, \widetilde{x}), \mathsf{tag}_f = (t_0, t_1, t_2)$. Let $\widetilde{C} := \mathsf{PRF}(\mathsf{key}, t_0) \oplus t_1$. Output 1 iff $\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = t_2$.

Theorem 4.3. *Assuming one-way functions, for any polynomials $n = n(\lambda), s = s(\lambda)$, there exists a functional tag system for the class $\mathcal{F}_\lambda = \mathcal{C}^{s,n}_\lambda$.*

Proof. We start by proving **dummy correctness**. Let f, x be arbitrary and let $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(x), \mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(f)$ with $\mathsf{tag}_x = (\mathsf{key}, \widetilde{x}), \mathsf{tag}_f = (t_0, t_1, t_2)$ and let $\widetilde{C} = \mathsf{PRF}(\mathsf{key}, t_0) \oplus t_1$. Since t_2 is uniformly random and independent of $\widetilde{C}, \widetilde{x}$, we have:

$$\Pr[\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 1] = \Pr[\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = t_2] = 2^{-\lambda}.$$

Next we prove **smart correctness**. Let f, x be arbitrary such that $f(x) = 1$ and let $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$, $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x), \mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$ with with $\mathsf{tag}_x = (\mathsf{key}, \widetilde{x}), \mathsf{tag}_f = (t_0, t_1 = \mathsf{PRF}(\mathsf{key}, t_0) \oplus \widetilde{C}, t_2)$. Then

$$\Pr[\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 1] = \Pr[\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = t_2] = 1$$

by the perfect correctness of garbled circuits.

Lastly, we prove the **security** of the functional tag system via a sequence of hybrid games where we change how the challenger generates input and function tags.

- Game_0: This is the game $\mathsf{FunTagGame}^0$ which outputs $\mathcal{A}^{\mathsf{DInputTag}(1^\lambda,\cdot),\mathsf{DFunctionTag}(1^\lambda,\cdot)}(1^\lambda)$, where the adversary $\mathcal{A}$ has the restrictions that: (1) it makes a single challenge input tag query x to the oracle $\mathsf{DInputTag}(1^\lambda, \cdot)$ and (2) all the queries f_i made to the $\mathsf{DFunctionTag}(1^\lambda, \cdot)$ oracle (both pre-challenge and post-challenge) satisfy $f_i(x) = 0$.
- Game_1: In this game, if the oracle $\mathsf{DFunctionTag}(1^\lambda, \cdot)$ ever samples a value t_0 that was already used in the response to a previous query, we define the output of the game to be 0.

  It is easy to see that Game_0 *and* Game_1 *are statistically indistinguishable since $t_0 \leftarrow \{0,1\}^\lambda$ is chosen randomly each time.*
- Game_2: In this game, we choose $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$ at the very beginning of the game and change the first oracle from $\mathsf{DInputTag}(1^\lambda, \cdot)$ to $\mathsf{SInputTag}(\mathsf{tsk}, \cdot)$.

  Game_1 *and* Game_2 *are identically distributed by the definition of* $\mathsf{DInputTag}, \mathsf{SInputTag}$ *and the fact that* tsk *is never used anywhere else.*
- Game_3: For all *pre-challenge function-tag queries*, switch to answering them using $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$ instead of $\mathsf{DFunctionTag}(\cdot)$. Indistinguishability follows via a sequence of internal hybrids $\mathsf{Game}^i_{2\to3}$ where the first i pre-challenge function-tag queries are answered using $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$ and the rest are answered using $\mathsf{DFunctionTag}(\cdot)$. Note that if the adversary makes q such queries then $\mathsf{Game}^0_{2\to3}$ is identical to Game_2 and $\mathsf{Game}^q_{2\to3}$ is identical to Game_3. To switch from $\mathsf{Game}^i_{2\to3}$ to $\mathsf{Game}^{i+1}_{2\to3}$ we introduce further sub-hybrids as follows:

1. $\mathsf{Game}^{i,1}_{2\to 3}$: At the very beginning of the game, when choosing $\mathsf{tsk} = (\mathsf{sk}, \mathsf{key})$ change from choosing the PRF key as $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$ to choosing

$$t_0 \leftarrow \{0,1\}^\lambda,\ \ \mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, t_0),\ \ r^* \leftarrow \{0,1\}^\ell,\ \ \mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, r^*).$$

   Then use the value t_0 to generate the i'th function-tag query (t_0, t_1, t_2).

   $\mathsf{Game}^{i}_{2\to 3}$ *is computationally indistinguishable from* $\mathsf{Game}^{i,1}_{2\to 3}$ *by SEPRF security.*

2. $\mathsf{Game}^{i,2}_{2\to 3}$: Choose t_0, eqKey at the beginning of the game as before, but do not choose r^*, key yet. For all function-tag queries before the i'th one, use eqKey instead of key to answer the query. When answering the i'th pre-challenge function-tag query (t_0, t_1, t_2), use the value t_0 sampled previously. When answering the challenge input-tag query later, choose $\widetilde{C} \leftarrow \{0,1\}^\ell$, $r^* \leftarrow t_1 \oplus \widetilde{C}$, $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, r^*)$.

   $\mathsf{Game}^{i,1}_{2\to 3}$ *is identically distributed to* $\mathsf{Game}^{i,2}_{2\to 3}$ *by the correctness of the SEPRF which says that* $\mathsf{PRF}(\mathsf{eqKey}, t'_0) = \mathsf{PRF}(\mathsf{key}, t'_0)$ *for all* $t'_0 \neq t_0$*, and if* $t'_0 = t_0$ *is ever chosen before the* i*'th query then the game outputs* 0 *in either case. Note that* r^* *is still uniform and independent of* t_1 *so defining* $r^* = t_1 \oplus \widetilde{C}$ *is the same as* $r^* \leftarrow \{0,1\}^\ell$*.*

3. $\mathsf{Game}^{i,3}_{2\to 3}$: When answering the challenge input-tag query, instead of choosing $\widetilde{C} \leftarrow \{0,1\}^\ell$, we now choose $u \leftarrow \{0,1\}^\lambda$, $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$.

   $\mathsf{Game}^{i,2}_{2\to 3}$ *is identically distributed to* $\mathsf{Game}^{i,3}_{2\to 3}$ *by the blindness property of blind garbled circuits and the fact that* $u \leftarrow \{0,1\}^\lambda$ *is chosen randomly.*

4. $\mathsf{Game}^{i,4}_{2\to 3}$: When answering the challenge input-tag query, instead of choosing $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$, we now choose $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$ where f_i is the function chosen in the i'th function-tag query.

   $\mathsf{Game}^{i,3}_{2\to 3}$ *is computationally indistinguishable from* $\mathsf{Game}^{i,4}_{2\to 3}$ *by the semi-adaptive simulation security of the garbled circuit. The reduction does not know the garbling key* sk *but is responsible for incorporating the equivocal PRF. It uses its oracle to* $\mathsf{GCircuit}(\mathsf{sk}, \cdot)$ *to answer all calls to* $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$*. During the challenge input-tag query for input* x*, the reduction hands* x *to its challenger to get* $\widetilde{x}$*. It then hands the challenger the circuit* $C^*_{f_i,u,t_2}$ *and gets* $\widetilde{C}$*. It uses the values* $\widetilde{x}, \widetilde{C}$ *to correctly answer the challenge input-tag query. If* $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$ *then the game is identical to* $\mathsf{Game}^{i,3}_{2\to 3}$ *and if* $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$ *then the game is identical to* $\mathsf{Game}^{i,4}_{2\to 3}$*. Here we rely on the fact that* $f_i(x) = 0$ *to ensure that* $C^*_{f_i,u,t_2}(x) = u$*.*

5. $\mathsf{Game}^{i,5}_{2\to 3}$: Instead of choosing $t_1 \leftarrow \{0,1\}^\lambda$ in the i'th function-tag query and then waiting to choose $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$, $r^* \leftarrow t_1 \oplus \widetilde{C}$, $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, r^*)$ in the input-tag query, we now choose $r^* \leftarrow \{0,1\}^\ell$, $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, r^*)$ at the very beginning of the game and then during the i'th function-tag query choose $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$

and set $t_1 = \widetilde{C} \oplus \mathsf{PRF}(\mathsf{key}, t_0)$. Furthermore, we now use key instead of eqKey to answer all function-tag queries before the i'th one.

$\mathsf{Game}_{2\to3}^{i.4}$ *is identically distributed to* $\mathsf{Game}_{2\to3}^{i.5}$. *The changes before the "furthermore" are just syntactic. In both cases* t_1, r^* *are random subject to* $t_1 \oplus r^* = \widetilde{C}$. *The "furthermore" part is identical by the correctness of the SEPRF which says that* $\mathsf{PRF}(\mathsf{eqKey}, t_0') = \mathsf{PRF}(\mathsf{key}, t_0')$ *for all* $t_0' \neq t_0$, *and if* $t_0' = t_0$ *is ever chosen before the* i*'th query then the game outputs* 0 *in either case. Recall this rule about outputting 0 when* t_0 *values are repeated was adopted in* Game_1.

6. $\mathsf{Game}_{2\to3}^{i+1}$: This is identical to $\mathsf{Game}_{2\to3}^{i.5}$, except that, instead of choosing

$$t_0 \leftarrow \{0,1\}^\lambda,\ \ \mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, t_0),\ \ r^* \leftarrow \{0,1\}^\ell,\ \ \mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, r^*)$$

at the beginning of the game we now just choose $\mathsf{key} \leftarrow \mathsf{KeyGen}(1^\lambda)$ at the very beginning and wait to choose t_0 until the i'th function-tag query.

$\mathsf{Game}_{2\to3}^{i.5}$ *is computationally indistinguishable from* $\mathsf{Game}_{2\to3}^{i+1}$ *by SEPRF security.*

Therefore the combination of the above hybrids shows that for each i: $\mathsf{Game}_{2\to3}^{i}$ is computationally indistinguishable from $\mathsf{Game}_{2\to3}^{i+1}$ and therefore Game_2 is computationally indistinguishable from Game_3.

- Game_4 For all *post-challenge function-tag queries*, switch to answering them using $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$ instead of $\mathsf{DFunctionTag}(\cdot)$. Indistinguishability follows via a sequence of internal hybrids $\mathsf{Game}_{3\to4}^{i}$ where the first i post-challenge function-tag queries are answered using $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$ and the rest are answered using $\mathsf{DFunctionTag}(\cdot)$. Note that if the adversary makes q such queries then $\mathsf{Game}_{3\to4}^{0}$ is identical to Game_3 and $\mathsf{Game}_{3\to4}^{q}$ is identical to Game_4. To switch from $\mathsf{Game}_{3\to4}^{i}$ to $\mathsf{Game}_{3\to4}^{i+1}$ we introduce further sub-hybrids as follows (essentially a simpler version of the sub-hybrids needed to go from Game_2 to Game_3 since we do not need to equivocate the SEPRF here):
  - $\mathsf{Game}_{3\to4}^{i.1}$: We change how the i'th post-challenge function-tag query is answered from choosing $t_1 \leftarrow \{0,1\}^\ell$ to choosing $u \leftarrow \lambda$, $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$ and setting $t_1 := \widetilde{C} \oplus \mathsf{PRF}(\mathsf{key}, t_0)$. We still choose $t_2 \leftarrow \{0,1\}^\lambda$ uniformly at random.

    $\mathsf{Game}_{3\to4}^{i}$ *is distributed identically to* $\mathsf{Game}_{3\to4}^{i.1}$ *by the blindness property of blind garbled circuits and the fact that* $u \leftarrow \{0,1\}^\lambda$ *is chosen randomly, which ensures that* $\widetilde{C}$ *is uniformly random over* $\{0,1\}^\ell$.

  - $\mathsf{Game}_{3\to4}^{i+1}$: We change how the i'th post-challenge function-tag query with function f_i is answered from $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$ to choosing $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$.

    $\mathsf{Game}_{3\to4}^{i.1}$ *is computationally indistinguishable from* $\mathsf{Game}_{3\to4}^{i+1}$ *by the semi-adaptive simulation security of the garbled circuit. The reduction does not know* sk. *During the challenge* *input-tag query for input x, the reduction hands x to its challenger to get $\widetilde{x}$ and uses it to answer the challenge input-tag query. It uses its oracle to $\mathsf{GCircuit}(\mathsf{sk}, \cdot)$ to answer all calls to $\mathsf{SFunctionTag}(\mathsf{tsk}, \cdot)$ aside from the i'th post-challenge function-tag after the input-tag query. For the i'th post-challenge function-tag query it picks the challenge circuit C^*_{f,u,t_2} and gets $\widetilde{C}$ from its oracle which it uses to answer that call. If $\widetilde{C} \leftarrow \mathsf{SimCircuit}(\mathsf{sk}, x, u)$ then the game is identical to $\mathsf{Game}^{i,1}_{3\to 3}$ and if $\widetilde{C} \leftarrow \mathsf{GCircuit}(\mathsf{sk}, C^*_{f_i,u,t_2})$ then the game is identical to $\mathsf{Game}^{i+1}_{3\to 4}$. Here we rely on the fact that $f_i(x) = 0$ to ensure that $C^*_{f_i,u,t_2}(x) = u$.*

Therefore the combination of the above hybrids shows that for each i: $\mathsf{Game}^{i}_{3\to 4}$ is computationally indistinguishable from $\mathsf{Game}^{i+1}_{3\to 4}$ and therefore Game_3 is computationally indistinguishable from Game_4.

- Game_5: This is the game $\mathsf{FunTagGame}^1$ which outputs $\mathcal{A}^{\mathsf{SInputTag}(1^\lambda,\cdot),\mathsf{SFunctionTag}(1^\lambda,\cdot)}(1^\lambda)$. This is the same as Game_4 except that we "undo" the change from Game_1: if the oracle $\mathsf{SFunctionTag}(1^\lambda, \cdot)$ ever samples a value t_0 that was already used in the response to a previous query, we continue as usual instead of defining the output of the game to be 0.

  It is easy to see that Game_4 and Game_5 are statistically indistinguishable since $t_0 \leftarrow \{0,1\}^\lambda$ is chosen randomly each time.

The above sequence of hybrids shows that $\mathsf{FunTagGame}^0$ and $\mathsf{FunTagGame}^1$ are computationally indistinguishable, which proves security. □

5 Adaptive ABE from WE via a Functional Tag System

We construct an adaptively secure ABE scheme for circuits using:

- a statistically binding commitment scheme $(\mathsf{Com.Setup}, \mathsf{Commit})$ per Definition 2.2,
- a statistically sound witness indistinguishable NIZK for NP $(\mathsf{NIZK.Setup}, \mathsf{Prove}, \mathsf{Verify})$ per Definition 2.4,
- a witness encryption scheme for NP $(\mathsf{WE.Enc}, \mathsf{WE.Dec})$ per Definition 2.6,
- a functional tag system for circuits $(\mathsf{DInputTag}, \mathsf{DFunctionTag}, \mathsf{SGen}, \mathsf{SInputTag}, \mathsf{SFunctionTag}, \mathsf{Trigger})$ per Definition 3.1 where the tag key $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$ is of length $|\mathsf{tsk}| = \ell(\lambda)$.

For any polynomials $s(\lambda), n(\lambda)$, let us fix the function class $\mathcal{C}^{s,n}_\lambda$ to consist of boolean circuits of size $s(\lambda)$ with $n(\lambda)$-bit input. We construct an ABE for the function class $\mathcal{C}^{s,n}_\lambda$ using a functional tag system for $\mathcal{C}^{s,n}_\lambda$ as a building block. We specify the NP relations $\mathsf{NIZK}.R, \mathsf{WE}.R$ for the NIZK and WE inside the construction.

Construction. The ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

- $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$: Choose $\mathsf{NIZK.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $\mathsf{Com.crs} \leftarrow \mathsf{Com.Setup}(1^\lambda)$, $r_0, r_1 \leftarrow \{0,1\}^\lambda$, and set

$$\mathsf{com}_0 := \mathsf{Commit}_{\mathsf{Com.crs}}(0; r_0), \mathsf{com}_1 := \mathsf{Commit}_{\mathsf{Com.crs}}(0^{\ell(\lambda)}; r_1).$$

  Output $\mathsf{mpk} := (\mathsf{Com.crs}, \mathsf{NIZK.crs}, \mathsf{com}_0, \mathsf{com}_1)$, $\mathsf{msk} := r_0$.

- $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$: Generate $\mathsf{tag}_f \leftarrow \mathsf{DFunctionTag}(1^\lambda, f)$. Give a NIZK proof

$$\pi \leftarrow \mathsf{Prove}_{\mathsf{NIZK.crs}}(\ \widetilde{x} = (\mathsf{Com.crs}, \mathsf{com}_0, \mathsf{com}_1, f, \mathsf{tag}_f)\ ,\ \widetilde{w} = r_0\ )$$

  for the NP relation

$$\mathsf{NIZK}.R = \left\{ (\widetilde{x}, \widetilde{w}) \ : \begin{array}{lcl} & \widetilde{x} = (\mathsf{Com.crs}, \mathsf{com}_0, \mathsf{com}_1, f, \mathsf{tag}_f) & \\ \text{either } \widetilde{w} = r_0 & : & \mathsf{com}_0 = \mathsf{Commit}_{\mathsf{Com.crs}}(0; r_0) \\ \text{or } \widetilde{w} = (\mathsf{tsk}, r_1, r_2) & : & \mathsf{com}_1 = \mathsf{Commit}_{\mathsf{Com.crs}}(\mathsf{tsk}; r_1) \\ & & \wedge \mathsf{tag}_f = \mathsf{SFunctionTag}(\mathsf{tsk}, f; r_2) \end{array} \right\}.$$

  Output $\mathsf{sk}_f = (f, \mathsf{tag}_f, \pi)$

- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, \mu)$: Generate $\mathsf{tag}_x \leftarrow \mathsf{DInputTag}(1^\lambda, x)$ and a witness encryption $\mathsf{WE.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \hat{x} = (\mathsf{Com.crs}, \mathsf{NIZK.crs}, x, \mathsf{com}_0, \mathsf{com}_1, \mathsf{tag}_x), \mu)$ for the relation

$$\mathsf{WE}.R = \left\{ (\hat{x}, \hat{w}) \ : \begin{array}{l} \hat{x} = (\mathsf{Com.crs}, \mathsf{NIZK.crs}, \mathsf{com}_0, \mathsf{com}_1, x, \mathsf{tag}_x), \hat{w} = (f, \mathsf{tag}_f, \pi) \\ \mathsf{Verify}_{\mathsf{NIZK.crs}}(\widetilde{x} = (\mathsf{Com.crs}, \mathsf{com}_0, \mathsf{com}_1, f, \mathsf{tag}_f), \pi) = 1 \\ \wedge f(x) = 1 \wedge \mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 0 \end{array} \right\}.$$

  Output $\mathsf{ct} = (x, \mathsf{tag}_x, \mathsf{WE.ct})$.

- $\mu := \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: Output $\mu := \mathsf{WE.Dec}(\mathsf{WE.ct}, (f, \mathsf{tag}_f, \pi))$.

Theorem 5.1. *Assuming witness encryption for NP, statistically sound NIZK for NP, statistically binding commitments and a functional tag system for circuits there exists an adaptively secure ABE for circuits.*

In particular, the above holds assuming witness encryption for NP, statistically sound NIZK for NP, and one-way functions. Alternately, the above holds assuming witness encryption for NP and any one of: (1) hardness of factoring, or (2) the decisional linear assumption in bilinear groups, or (3) the learning with errors (LWE) assumption. Lastly, the above holds just assuming evasive LWE.

Proof. We show that the construction given above is an adaptively secure ABE for $\mathcal{C}_\lambda^{s,n}$ assuming the security of the components. The correctness of the ABE follows from the correctness of the WE and NIZK along with correctness (property 1) of the functional tag system. To prove adaptive security, we define a sequence of games:

- Game_0^b: This is the ABE game $\mathsf{ABEGame}^b$ between the adversary and the challenger.

- Game_1^b: We modify the game so that the challenger initially chooses a "smart tag system key" $\mathsf{tsk} \leftarrow \mathsf{SGen}(1^\lambda)$. When answering key queries, the challenger now samples keys $\mathsf{sk}_f = (f, \mathsf{tag}_f, \pi)$ by choosing a "smart function tag" $\mathsf{tag}_f \leftarrow \mathsf{SFunctionTag}(\mathsf{tsk}, f)$ instead of a dummy one. For the challenge ciphertext $\mathsf{ct} = (x, \mathsf{tag}_x, \mathsf{WE.ct})$, the challenger now chooses a "smart input tag" $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$ instead of a dummy one. The keys and the challenge ciphertext are otherwise generated the same way as previously.

Game_0^b *and* Game_1^b *are indistinguishable by the security property (property 3) of the functional tag system. Note that in the ABE adaptive security game, the adversary can only choose attribute* x *such that* $f_i(x) = 0$ *for all key queries* f_i*, which matches the restriction on the adversarial queries of the functional tag system.*

- Game_2^b: We modify the game so that, when choosing $\mathsf{mpk} = (\mathsf{Com.crs}, \mathsf{NIZK.crs}, \mathsf{com}_0, \mathsf{com}_1)$, the challenger sets $\mathsf{com}_1 := \mathsf{Commit}_{\mathsf{Com.crs}}(\mathsf{tsk}; r_1)$ to be a commitment to tsk instead of $0^{\ell(\lambda)}$.

  Game_1^b *and* Game_2^b *are indistinguishable by the computational hiding security of the commitment scheme. Note that the commitment randomness* r_1 *does not appear anywhere else in the game.*

- Game_3^b: We modify how the challenger answers key queries with keys $\mathsf{sk}_f = (f, \mathsf{tag}_f, \pi)$. In particular, the challenger now generates the proof π as:

$$\pi \leftarrow \mathsf{Prove}_{\mathsf{NIZK.crs}}(\ \widetilde{x} = (\mathsf{Com.crs}, \mathsf{com}_0, \mathsf{com}_1, f, \mathsf{tag}_f)\ ,\ \widetilde{w} = (\mathsf{tsk}, r_1, r_2)\ )$$

  using the witness $\widetilde{w} = (\mathsf{tsk}, r_1, r_2)$ where r_2 is the randomness used to generate $\mathsf{tag}_f := \mathsf{SFunctionTag}(\mathsf{tsk}, f; r_2)$, instead of using the witness $\widetilde{w} = r_0$.

  Game_2^b *and* Game_3^b *are indistinguishable by witness indistinguishability security of the NIZK.*

- Game_4^b: In this game, when choosing the master public key $\mathsf{mpk} = (\mathsf{Com.crs}, \mathsf{NIZK.crs}, \mathsf{com}_0, \mathsf{com}_1)$, the challenger now sets $\mathsf{com}_0 := \mathsf{Commit}_{\mathsf{Com.crs}}(1; r_1)$ to be a commitment to 1 instead of 0.

  Game_3^b *and* Game_4^b *are indistinguishable by the computational hiding security of the commitment scheme. Note that the commitment randomness* r_0 *does not appear anywhere else in the game.*

- Game_5^b: In this game, when choosing the challenge ciphertext $\mathsf{ct} = (x, \mathsf{tag}_x, \mathsf{WE.ct})$, the challenger samples

$$\mathsf{WE.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \hat{x} = (\mathsf{Com.crs}, \mathsf{NIZK.crs}, x, \mathsf{com}_0, \mathsf{com}_1, \mathsf{tag}_x), 0)$$

  to be an encryption of 0 rather than the bit b.

  Game_4^b *and* Game_5^b *are indistinguishable by WE security. Firstly, note that whenever* Com.crs *is binding and* NIZK.crs *is sound (see Definitions 2.2 and 2.4) then the statement* $\hat{x}$ *is false. To see this, assume otherwise that* $(\hat{x}, \hat{w}) \in \mathsf{WE}.R$ *for some* $\hat{w} = (f, \mathsf{tag}_f, \pi)$*. Then it must hold that* $f(x) = 1$*,* $\mathsf{Trigger}(\mathsf{tag}_f, \mathsf{tag}_x) = 0$ *and* $\mathsf{Verify}_{\mathsf{NIZK.crs}}(\widetilde{x} = (\mathsf{Com.crs}, \mathsf{com}_0, \mathsf{com}_1, f, \mathsf{tag}_f), \pi) = 1$*. The latter implies that there exists some* $\widetilde{w}$ *such that* $(\widetilde{x}, \widetilde{w}) \in \mathsf{NIZK}.R$*. Since* $\mathsf{com}_0 = \mathsf{Commit}_{\mathsf{Com.crs}}(1; r_0)$, $\mathsf{com}_1 = \mathsf{Commit}_{\mathsf{Com.crs}}(\mathsf{tsk}; r_1)$ *this in turn implies that* $\mathsf{tag}_f = \mathsf{SFunctionTag}(\mathsf{tsk}, f; r_2)$ *for some* r_2*. But since* $\mathsf{tag}_x \leftarrow \mathsf{SInputTag}(\mathsf{tsk}, x)$*, the above contradicts property 2 of the functional tag system. Secondly, note that* Com.crs *is binding and* NIZK.crs *is sound with overwhelming probability, and therefore the statement* $\hat{x}$ *is false with overwhelming probability. Given the above, an adversary distinguishes* Game_4^b *and* Game_5^b *with non-negligible probability must distinguish* $\mathsf{WE.Enc}(1^\lambda, \hat{x}, 0)$ *and* $\mathsf{WE.Enc}(1^\lambda, \hat{x}, 1)$ *for a false statement* $\hat{x}$ *with non-negligible probability.*

Note that $\mathsf{Game}_5^0 \equiv \mathsf{Game}_5^1$ since the game completely ignores the bit b. Therefore, by the hybrid argument, we have Game_0^0 is indistinguishable from Game_0^1, which implies ABE seucrity. □

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 657–677. Springer, Heidelberg, August 2015. 2, 3

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, Heidelberg, August 2004. 2

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Heidelberg, August 2001. 2

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, Heidelberg, May 2014. 2, 4

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 776–806. Springer, Heidelberg, August 2020. 3

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564. Springer, Heidelberg, April / May 2018. 4, 5, 11

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 363–384. Springer, Heidelberg, August 2016. 2

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 1082–1090. ACM Press, June 2019. 8

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, Heidelberg, May 2003. 8

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569. Springer, Heidelberg, August 2017. 2

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE Computer Society Press, October 1990. 8

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 498–531. Springer, Heidelberg, August 2023. 3

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, Heidelberg, May / June 2006. 2

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476. ACM Press, June 2013. 2, 3

[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 Annual ACM Symposium on Theory of Computing*, pages 555–564. ACM Press, June 2013. 1

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 361–388. Springer, Heidelberg, October / November 2016. 2

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 661–695. Springer, Heidelberg, August 2017. 3

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, Heidelberg, May / June 2006. 8

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on*

Computer and Communications Security, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. 1

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554. ACM Press, June 2013. 2, 4

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178. Springer, Heidelberg, August 2016. 5, 8, 9

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73. ACM Press, June 2021. 3

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, Heidelberg, May / June 2010. 2, 3

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 58–76. Springer, Heidelberg, May 2014. 2

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. 7

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 422–439. Springer, Heidelberg, March 2012. 1

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, Heidelberg, August 2019. 8

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005. 2

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, Heidelberg, August 1984. 2

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, Heidelberg, May 2005. 1

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 62–85. Springer, Heidelberg, August 2019. 2

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 535–559. Springer, Heidelberg, August 2022. 3

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 195–221. Springer, Heidelberg, December 2022. 3

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, Heidelberg, May 2005. 2

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, Heidelberg, August 2009. 2, 3

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 678–697. Springer, Heidelberg, August 2015. 2, 3