



Static vs. Adaptive Security in Perfect MPC: A Separation and the Adaptive Security of BGW

Gilad Asharov ✉ 

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Ran Cohen ✉ 

Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Oren Shochat ✉

Department of Computer Science, Bar-Ilan University, Ramat-Gan Israel

Abstract

Adaptive security is a highly desirable property in the design of secure protocols. It tolerates adversaries that corrupt parties as the protocol proceeds, as opposed to static security where the adversary corrupts the parties at the onset of the execution. The well-accepted folklore is that static and adaptive securities are *equivalent* for perfectly secure protocols. Indeed, this folklore is backed up with a transformation by Canetti et al. (EUROCRYPT'01), showing that any perfectly secure protocol that is statically secure and satisfies some basic requirements is also adaptively secure. Yet, the transformation results in an adaptively secure protocol with *inefficient simulation* (i.e., where the simulator might run in super-polynomial time even if the adversary runs just in polynomial time). Inefficient simulation is problematic when using the protocol as a sub-routine in the computational setting.

Our main question is whether an alternative *efficient* transformation from static to adaptive security exists. We show an inherent difficulty in achieving this goal generically. In contrast to the folklore, we present a protocol that is perfectly secure with efficient static simulation (therefore also adaptively secure with inefficient simulation), but for which efficient adaptive simulation does not exist (assuming the existence of one-way permutations).

In addition, we prove that the seminal protocol of Ben-Or, Goldwasser and Wigderson (STOC'88) is secure against adaptive, semi-honest corruptions with *efficient* simulation. Previously, adaptive security of the protocol, as is, was only known either for a restricted class of circuits, or for all circuits but with inefficient simulation.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases secure multiparty computation, perfect security, adaptive security, BGW protocol

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.15

Related Version *Full Version:* <https://eprint.iacr.org/2022/758.pdf>

Funding *Gilad Asharov:* Sponsored by the Israel Science Foundation (grant No. 2439/20), by JPM Faculty Research Award, by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

Ran Cohen: Research partially supported by NSF grant no. 2055568.

Oren Shochat: Sponsored by the Israel Science Foundation (grant No. 2439/20).

1 Introduction

Secure multiparty computation (MPC) [31, 20] enables a set of mutually distrustful parties to compute a joint function while keeping the privacy of their inputs and the correctness of their outputs. The seminal results from the '80s [31, 20, 5, 11, 30] as well as the vast majority



© Gilad Asharov, Ran Cohen, and Oren Shochat;
licensed under Creative Commons License CC-BY 4.0
3rd Conference on Information-Theoretic Cryptography (ITC 2022).

Editor: Dana Dachman-Soled; Article No. 15; pp. 15:1–15:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of MPC protocols in the literature were proven secure with respect to a *static* adversary; that is, security is guaranteed as long as the adversary decides which parties to corrupt at the onset of the execution. A more realistic setting, first considered by Beaver and Haber [4] and by Canetti et al. [10], considers an *adaptive* adversary, who can dynamically decide which parties to corrupt during the course of the protocol. Adaptive security is known to be strictly stronger than static security with many impossibility results separating the two notions, e.g., [10, 9].

In this work we focus on the well-studied setting of *perfect security*, where all existing separations from the literature no longer hold. Perfectly secure protocols guarantee security facing computationally unbounded adversaries without any error probability. This is a highly desirable property that in some sense provides the strongest security notion for MPC. For example, Kushilevitz et al. [24] showed that any perfectly secure protocol that is proven secure in the standalone model using a straight-line and black-box simulation¹ automatically guarantees security under universal composition (UC) [8].² Most relevant to our work, Canetti et al. [9] showed that *any* perfect statically secure protocol, satisfying basic requirements, remains secure also in the presence of adaptive adversaries. This result, however, comes with a caveat, since the transformation from static to adaptive security does not preserve the efficiency of the simulation.

Roughly speaking, a protocol is deemed secure if any attack on an execution of the protocol in the real world can be simulated also in an ideal world where a trusted party receives the inputs from all the parties and computes the function on their behalf. It is desirable that the simulator, i.e., the adversary who simulates the attack in the ideal world, will use roughly the same resources as the adversary who carries out the attack on the real-world protocol. In particular, we would like the simulator to run in polynomial time with respect to the running time of the real-world adversary; if so, we say that the simulation is *efficient*.

Unfortunately, the adaptive simulator in [9] does *not* run in polynomial time with respect to the real-world adversary. This means that given a perfectly secure protocol against static adversaries with an efficient simulator, the transformation in [9] guarantees adaptive security, albeit with an inefficient simulator. This leads us to the following fundamental question:

*Does perfect, static security with efficient simulation imply
perfect, adaptive security with efficient simulation?*

Stated differently, is there another generic transformation from static to adaptive security, other than [9], that preserves the efficiency of the simulation? Are there other assumptions on the structure of the protocol and/or on the static simulation that might lead to such an efficient transformation? Or, perhaps, do there exist protocols that are statically secure but for which *efficient* adaptive simulation simply does not exist?

Efficient vs. inefficient simulation. One may ask whether the efficiency loss in the simulation makes a difference when considering perfect security: If security is anyway guaranteed against computationally unbounded adversaries, does it matter if the simulator is inefficient? Indeed, inefficient simulation is a weaker-yet-acceptable security notion when considering information-theoretic security.

¹ A simulator is called *straight-line* if it does not rewind the adversary, and is called *black-box* if it does not rely on the code of the adversary.

² We note that Backes et al. [3] showed that this transformation no longer holds if the simulator is not straight-line.

It turns out that inefficient simulation has an undesirable impact when considering *composition* of secure protocols. For example, consider a perfectly secure protocol π for computing a function f that is defined over secure communication channels, and consider a computationally secure realization of secure channels over authenticated channels (e.g., using non-committing encryption [10]). If π is secure with efficient simulation, then a composition theorem (e.g., from [7, 8]) can be used to derive a computationally secure protocol for f over authenticated channels. However, if the simulation is *inefficient* then the composition will not go through since it will result with a super-polynomial time adversary that can break the cryptographic assumptions used to realize secure channels.

A case study: on the adaptive security of the BGW protocol. The seminal results of Ben-Or, Goldwasser, and Wigderson [5] and of Chaum, Crépeau, and Damgård [11] show that any function f can be computed by an n -party protocol with perfect security as long as $t < n/2$ of the parties are corrupted in the semi-honest setting, and as long as $t < n/3$ are corrupted in the malicious setting. A full proof of security for the BGW protocol was given in 2011 by Asharov and Lindell [1, 2]. The proof was specified in the static, standalone setting, and universally composable security and security against adaptive adversaries were derived using the transformations of Kushilevitz et al. [24] and Canetti et al. [9], respectively. However, as discussed above, adaptive security is obtained with an inefficient simulation.

This issue was revisited by Damgård and Nielsen [15], who showed that the semi-honest version of the BGW protocol achieves adaptive security with efficient simulation. However, the result of [15] holds only for circuits in which each output wire is a direct output of a multiplication gate. Obviously, one can manually add such “multiplications with 1” to each output wire. While this seems suffice, there are two reasons why we revisit this problem: First, for linear functions, the semi-honest version of the BGW protocol can tolerate up to n corruptions, whereas the requirement that each output wire is a direct output of a multiplication gate reduces the corruption threshold to $t < n/2$. Second, this subtlety has been neglected by prior works that relied on [15], e.g., Lin et al. [26, Lem. 6.2] claimed that any degree-2 function can be computed by the BGW protocol in two rounds with adaptive security and efficient simulation. Yet, when adding multiplication gates the round complexity increases, which implies a gap in the literature as there is no proof for the claim mentioned in [26].

Alternatively, one can interpret this additional restriction on the circuit as adding some re-randomization step at the very end of the protocol before the parties reconstruct their output. Is this step essential to achieve adaptive security for all circuits? Can one prove the adaptive security of the original protocol directly, without the additional communication round?

1.1 Our Results

Our work revisits the question of static versus adaptive in perfectly secure multiparty computation. We show that in contrast to the “weaker” definition of adaptive security (i.e., inefficient simulation), perfect static security no longer implies perfect adaptive security when demanding the simulation to be efficient, even when merely considering semi-honest adversaries. Complementarily, we show that the BGW protocol is adaptively secure with efficient simulation even without changing the underlying circuit; this answers an open question posed by Damgård and Nielsen [15]. We focus on semi-honest security, which is

15:4 Static vs. Adaptive Security in Perfect MPC

enough to highlight the subtleties that arise; indeed, the gap in the literature discussed above already appears in this setting. We conjecture that the analysis extends to the malicious case using standard secret-sharing techniques.

We proceed to describe our results in more details.

Separating perfect adaptive security from perfect static security. Our first result shows that under some cryptographic assumptions, there is no hope of finding an alternative (efficient) transformation to that of Canetti et al. [9], and that inefficiency of the adaptive simulation is inherent. More precisely, that there exist protocols that admit perfect, static security with efficient simulation but for which an efficient adaptive simulation does not exist.

► **Theorem 1.** *Assume the existence of a one-way permutation. Then, there exists an n -party functionality f and a protocol that securely computes f with efficient perfect static simulation, but for which efficient perfect adaptive simulation does not exist.*

The theorem is proven by showing a protocol for which all the additional requirements of [9] hold and therefore (inefficient) adaptive simulation *does* exist for this protocol, but an efficient adaptive simulator can be used to invert the one-way permutation with inverse-polynomial probability. Interestingly, this implies that the protocol is regarded as adaptively secure in the perfect setting, but the exact same protocol is not adaptively secure in the *computational* setting, where both the adversary and the simulator should run in polynomial time in the security parameter. We therefore derive the following somewhat counter-intuitive corollary.

► **Corollary 2.** *Assume the existence of a one-way permutation. Then, there exists an n -party functionality f and a protocol that securely computes f with perfect adaptive security, but that does not securely compute f with computational adaptive security.*

Revisiting adaptive security in the standalone setting. The above does not rule out the possibility of finding additional requirements from the protocol that would imply efficient adaptive simulation. This is exactly the approach taken by Damgård and Nielsen [15], who showed that under additional requirements of the protocol, an efficient adaptive simulation exists.

The transformation of [15] is directly proved in the UC framework with its full generality, capturing reactive functionalities and concurrency issues at once. However, the strong guarantees do not come without a price, since the requirements from the static simulator must capture multiple input phases (as required for reactive computations) and deal with the technical overhead needed for concurrent composition, e.g., incorporating an “online” environment to the definition. As a small side contribution we simplify this transformation.

Specifically, in addition to proving perfect static security, the transformation of [15] requires an additional (efficient!) algorithm, called “Patch,” for sampling randomness that explains the simulated protocol whenever a corruption occurs. The adaptive simulator invokes the static simulator on a dynamically growing set of corrupted parties (initially empty). At any point, the simulation of the protocol towards the adaptive adversary is done by forwarding messages from the adaptive adversary to the static simulator, and vice-versa. Upon a corruption of a new party, say of P_i , the Patch algorithm receives the state of the static simulator until this point together with the input and output of P_i , and outputs a new state for the static simulator that allows the continuation of the simulation as if P_i was statically corrupted from the beginning.

We then propose an alternative recipe for proving universal composability and (efficient) adaptive security in the perfect setting:

1. Prove that the protocol is (efficiently) statically secure in the perfect *standalone* setting and that the protocol satisfies some natural requirements (similar to those in [9]).
2. Show the existence of an efficient “Patch” algorithm corresponding to the static simulator.
3. We show a transformation (which is essentially a distilled version of [15]) that the protocol is perfectly adaptively secure with an efficient simulator in the standalone setting.
4. Using [24], the protocol is also secure in the perfect adaptive setting with efficient simulation and with universal composability.

We remark that this result is not technically novel and is inspired by [15]. We hope that providing an alternative definition in the standalone setting would simplify proofs of efficient adaptive security in the future, as the designer of the protocol can focus on the standalone setting.

Adaptive security of the BGW protocol. Finally, we follow our recipe proposed above and show that the (semi-honest) BGW protocol is efficiently adaptively secure. No additional step to the protocol is needed, and the proof works for any circuit (as opposed to the proof of [15]). This result solves an open problem raised by [15], whether the assumption on the circuit (that each output wire is a direct output of a multiplication gate) is necessary. This reaffirms the security of the BGW protocol [5], and closes a gap in the proofs of [1, 15], providing sound foundations for cryptography.

► **Theorem 3.** *Let f be a deterministic n -party functionality. The BGW protocol securely computes f with perfect adaptive security and efficient simulation facing a semi-honest adversary corrupting $t < n/2$ parties.*

Further, if f is a linear function, the BGW protocol securely computes f with perfect adaptive security and efficient simulation facing a semi-honest adversary corrupting $t < n$ parties.

1.2 Related Work

Adaptive security is known to be strictly stronger than static security in many settings, with many impossibility results separating the two notions. Below, we compare our separation to existing separations between static and adaptive security from the literature.

The first separation was presented by [10] and relied on a positive error probability of the statically secure protocol. They considered a dealer that secret shares a value to a random set of parties and later announces their identities; an adaptive adversary can corrupt the members of the set and learn the value while a static adversary can only guess this set ahead of time and succeed with negligible (yet positive) probability. In this setting, the seminal protocol of Rabin and Ben-Or [30] that guarantees statistical information-theoretic security is not secure in the adaptive setting, as illustrated by Cramer et al. [14], who gave an adaptively secure variant of the protocol. Separations based on statistical security are different than ours as we consider *perfect* protocols that have zero error probability.

In the computational-security setting, many statically secure primitives do not remain secure under adaptive corruptions. For example, Nielsen [29] showed that public-key encryption for unbounded messages requires a programmable random oracle. Canetti et al. [9] separated adaptively secure commitments from statically secure ones. Lindell and Zarusim [27] showed that achieving adaptively secure oblivious transfer (OT) requires stronger assumptions than statically secure OT. Katz et al. [23] ruled out adaptively secure fully homomorphic

encryption; the latter result was generalized in [13]. Separations based on computational assumptions are different than ours as we consider *information-theoretic* protocols that remain secure against computationally unbounded adversaries.

Canetti et al. [9] showed a separation based on the ability of the adversary to corrupt a party and change its input to the protocol based on messages that have already been transmitted. Similar separations were illustrated also for broadcast protocols [22, 12]. Canetti et al. [9] showed that such separations no longer hold when considering protocols that have a *committal round* [28], i.e., a fixed round in which all inputs to the protocol are committed. These separations hold only for malicious adversaries that can deviate from the protocol; our separation applies for semi-honest adversaries that cannot deviate from the protocol in any way.

Other separations are known when considering restricted interaction patterns, as was shown by Garay et al. [17] for protocols with sublinear communication, and by Boyle et al. [6] for protocols whose communication graph admits a sublinear cut. Our separation relies on the BGW protocol that induces a complete communication graph.

Finally, Garg and Sahai [18] showed that constant-round MPC with black-box simulation in the plain model cannot tolerate corruption of all of the parties. Our result holds irrespective of the number of rounds and does not require corrupting all the parties.

Organization of the paper. In Section 2 we present a technical overview of our results, in Section 3 the preliminaries, and in Section 4 we prove our separation result, showing the existence of a protocol that has inefficient adaptive simulation but no efficient adaptive simulation (assuming one-way permutations exist). Due to space limitation we omit the definition of secure multiparty computation and the proof of adaptive security of the BGW protocol, and refer the reader to the full version of this paper.

2 Technical Overview

We provide a technical overview of our results. In Section 2.1 we review our separation result, showing the existence of a protocol that has inefficient adaptive simulation but no efficient adaptive simulation (under some cryptographic assumptions). In Section 2.2, we review the adaptive security of the BGW protocol.

2.1 Static Security Does Not Imply Adaptive Security

Definition of adaptive security. We first recall the definition of adaptive security. We remark that since the transformation of [9] assumes some additional properties from the statically secure protocol and its simulation (specifically, it assumes that the protocol has a straight-line, black-box simulation), our description here incorporates those properties in the informal definition.

Given a protocol π , an adaptive adversary might corrupt parties on the fly as the protocol proceeds. Upon corruption, the adversary sees the corrupted party's random tape, the input it used, and all the messages it received so far. From that point on, the adversary completely controls the behavior of P_i . As the protocol proceeds, the adversary might decide to corrupt additional parties, as a function of whatever it saw so far.

We follow the ideal/real simulation paradigm and say that a protocol is adaptively secure if for every such an adversary in the real world, there exists a simulator in the ideal world that simulates its behavior. In the ideal world, the simulator invokes the real-world adversary and simulates the honest parties sending their message to the corrupted parties (without knowing

the inputs of the honest parties). At every round, the adversary might ask to corrupt some party P_i in the simulated protocol, and the simulator corrupts the corresponding party in the ideal world. Upon corruption, the simulator learns the input of P_i (and its output, if it was already computed by the trusted party), and it has to provide the adversary the input *together with randomness* that explains the messages sent by this party in the simulation until this point (known also as “equivocality” in the literature).

This is the challenging part of the proof as the simulator has to first simulate P_i without knowing its input, “commit” to some messages on its behalf, and later upon corruption find a randomness r_i that explains all the messages that were sent so far according to the input x_i . Note that the simulator is not allowed to rewind the adversary at any point of the execution, i.e., the simulation is “straight-line.”

First attempt. The transformation of [9] shows that for any perfectly secure protocol, randomness r_i describing the behavior of P_i so far, exists. This implies that the simulator can *always* find it; however, finding it might not be an efficient procedure. Our first attempt is adding some cryptographic hardness while targeting exactly this procedure of finding the matching randomness. That is, our goal is making the finding of the randomness a computationally hard problem.

It is intuitive to simply take the BGW protocol, even just for semi-honest security and for computing a linear function, with one modification: Before a party starts the protocol, it takes its random tape r , and uses $\text{OWP}(r)$ as its randomness in the protocol, where OWP is a one-way permutation. The intuition is that whenever the adversary asks to corrupt some party, the simulator would effectively have to invert the one-way permutation, which is computationally infeasible. The construction is still statically secure since the static simulator only goes “forward”; that is, it chooses some randomness r and then uses $\text{OWP}(r)$ as the randomness of the simulated honest party. Moreover, an efficient simulation exists since r exists; however, it seems that an adaptive adversary will have to move “backward” and invert the one-way permutation.

To elaborate further, let n be the number of parties, and consider the function $f(a_1, \dots, a_n) = \sum_{i=1}^n a_i$; that is, all parties receive the same output, which is the sum of their inputs. We consider some finite field \mathbb{F} with $|\mathbb{F}| > n$. The protocol is as follows:

- **Input:** P_i has $a_i \in \mathbb{F}$ as input, and randomness r_i .
- **The protocol:**
 1. P_i computes $(r_{i,1}, \dots, r_{i,n-1}) = \text{OWP}(r)$, where each $r_{i,j} \in \mathbb{F}$.
 2. P_i secret shares its input a_i using Shamir’s secret sharing scheme with degree $n - 1$, using the polynomial $h_i(x) = a_i + r_{i,1}x + \dots + r_{i,n-1}x^{n-1}$. It gives to each party P_j privately a point $h_i(\alpha_j)$.
 3. Upon receiving $h_1(\alpha_i), \dots, h_n(\alpha_i)$ from all the parties, the party P_i computes $\beta_i = \sum_{j=1}^n h_j(\alpha_i)$ and sends β_i to all other parties.
 4. Upon receiving β_1, \dots, β_n , each party reconstructs the unique polynomial $H(x)$ for which it holds for every α_j that $H(\alpha_j) = \beta_j$, and outputs $H(0)$.

Simulating this protocol. The above protocol can be simulated for every $t < n$ parties. Consider an adaptive simulator, and assume that the adversary already corrupted $n - 2$ parties on the onset of the execution, say P_3, \dots, P_n . The simulator then simulates just two honest parties: P_1 and P_2 . As it does not know their inputs it just gives the adversary $2 \cdot (n - 2)$ independent random points as the messages of the two honest parties it is simulating. For concreteness, assume that the messages the simulated P_1 sent are $(\gamma_3, \dots, \gamma_n)$, where γ_i

is supposed to be delivered to P_i . Now, assume that the adversary asks to corrupt P_1 and that the simulator receives its input a_1 . The simulator can now find a random polynomial $h_1(x)$ of degree $n - 1$ under the constraints that

$$h_1(0) = a_1, \quad \text{and} \quad h_1(\alpha_3) = \gamma_3 \quad \dots \quad h_1(\alpha_n) = \gamma_n .$$

Note that these constraints define overall $n - 1$ points. Since $h_1(x)$ has n coefficients, there are $|\mathbb{F}|$ different polynomials that satisfy these $n - 1$ constraints. The simulator can pick one more point at random, and then uniquely determine the polynomial $h_1(x)$ using interpolation. For that particular polynomial, the simulator has to invert the one-way permutation and find the corresponding randomness.

The problem – the reduction to OWP. It is hard to use the above adaptive simulator to construct an inverter for the one-way permutation on some challenge point $y^* \in |\mathbb{F}|^{n-1}$, corresponding to the $n - 1$ leading coefficients, say $y^* = (r_{1,1}, \dots, r_{1,n-1})$. The problem is that once the adaptive simulator “commits” to the values $(\gamma_3, \dots, \gamma_n)$, it might be the case that there is *no* input a_1 for which the polynomial $h_1(x) = a_1 + r_{1,1}x + \dots + r_{1,n-1}x^{n-1}$ satisfies the constraints $h_1(\alpha_3) = \gamma_3, \dots, h_1(\alpha_n) = \gamma_n$. That is, there is no input a_1 in the support that agrees with the challenge y^* and with the simulated view, simultaneously.

Second attempt. To solve the above technicality, we change the functionality and the protocol. Instead of having the input of each party P_i be just $a_i \in \mathbb{F}$, it is augmented to be a polynomial $g_i(x)$ of degree $n - 1$ over \mathbb{F} . The functionality is then defined as:

$$f(g_1(x), \dots, g_n(x)) = \sum_{i=1}^n g_i(0).$$

Note that the parties input polynomials, while all the leading coefficients do not affect the output of the computation. In the protocol, each party P_i then invokes $(r_{i,1}, \dots, r_{i,n-1}) = \text{OWP}(\rho_i)$ where ρ_i is its random tape, and then defines the polynomial $h_i(x) := g_i(x) + r_{i,1}x + \dots + r_{i,n-1}x^{n-1}$. It then sends to each other party P_j the point $h_i(\alpha_j)$, just as in Step 2 in the previous protocol.

The difference is that now, no matter what points $(\gamma_3, \dots, \gamma_n)$ the adversary commits to as the messages P_1 had sent, for any challenge $y^* = (r_1^*, \dots, r_{n-1}^*)$, the inverter of the OWP can choose an input $g_1(x)$ such that the polynomial $g_1(x) + r_1^*x + \dots + r_{n-1}^*x^{n-1}$ is in the support of the polynomials that the adaptive simulator might choose. To see that, given a challenge $y^* = (r_1^*, \dots, r_{n-1}^*)$ to the inverter, and given $(\gamma_3, \dots, \gamma_n)$ that were chosen by the simulator as the messages P_1 had sent in the first round, the inverter chooses two additional points γ_1 and γ_2 at random. It then interpolates the unique polynomial $h(x)$ such that for every $i \in [n]$ it holds that $h(\alpha_i) = \gamma_i$. It computes the polynomial $g_1(x) = h(x) - (r_1^*x + \dots + r_{n-1}^*x^{n-1})$, and gives $g_1(x)$ to the adaptive simulator as the input of P_1 .

The simulator now has to reply with some randomness ρ' for which $(r'_{1,1}, \dots, r'_{1,n-1}) = \text{OWP}(\rho')$ such that $h'_1(x) = g_1(x) + \sum_{k=1}^{n-1} r'_{1,k}x^k$ and it holds that $h'_1(\alpha_3) = \gamma_3, \dots, h'_1(\alpha_n) = \gamma_n$. Since $h'_1(\alpha_1)$ and $h'_2(\alpha_2)$ are not determined, the simulator essentially has $|\mathbb{F}|^2$ different polynomials to choose from. However, *one of them* corresponds to the challenge y^* . Moreover, y^* is distributed uniformly in the support of all valid solutions for the simulator. Since the adaptive simulator simulates with perfect security, the inverter succeeds in inverting y^* with probability $1/|\mathbb{F}|^2$. By tuning the parameters (such that $|\mathbb{F}|$ is polynomial in the security parameter), this is an inverse-polynomial advantage. Assuming that OWP is a one-way permutation, this implies that the adaptive simulator cannot run in polynomial time.

2.2 Adaptive Security of the BGW Protocol

The BGW Protocol. We briefly recall the semi-honest version of the BGW protocol [5] that is secure for $t < n/2$ corruptions; see [1] for more details.

Input sharing: Initially, each party P_i secret shares its input x_i using a $(t + 1)$ -out-of- n Shamir secret sharing (t is the privacy threshold and $t + 1$ are needed for reconstruction), and sends the j^{th} share to P_j . Specifically, P_i chooses a random polynomial q_i of degree (at most) t with constant term x_i , and sends to P_j the share $q_i(\alpha_j)$.

Circuit emulation: The parties evaluate the circuit gate by gate on their shares. Addition and scalar-multiplication gates are evaluated locally; that is, each party performs the operation on its shares. Multiplication requires a dedicated sub-protocol in which each party inputs shares for two values a and b , and obtains a share for the product ab as output. For simplicity of the technical overview, we consider that this operation is done using the help of a trusted party (i.e., we work in the f_{mult} -hybrid model).

Output reconstruction: Upon conclusion of the circuit emulation, each party holds one share for each output wire. For simplicity, assume that output wire o_i is the private output for P_i . Then, each party P_j sends the relevant share $\beta_{j \rightarrow i}$ for P_i , who can reconstruct the polynomial g_i interpolating the points $(\alpha_1, \beta_{1 \rightarrow i}), \dots, (\alpha_n, \beta_{n \rightarrow i})$, and obtain $g_i(0)$ as its output.

Static simulation. To simulate the view of the adversary for the set of corrupted parties I of cardinality at most t , the static simulator simulates the input-sharing phase by choosing randomness for the corrupted parties (which, in turn, determines the polynomials they use in the input-sharing phase). Moreover, for the honest parties, the simulator just chooses $|I|$ random elements for each honest party and simulates the honest parties sending their shares on their inputs to the adversary. The simulator can then locally compute the shares of the corrupted parties on each internal wire of the circuit, while for simulating an invocation of f_{mult} , the simulator again just provides the adversary with $|I|$ random shares. In the output-reconstruction phase, the simulator has to provide all shares on the output wires of corrupted parties. It knows the constant term of each such wire, and it knows $|I|$ shares on each wire, and therefore it is possible to interpolate a random polynomial that passes through the $|I| \leq t$ shares and the known constant term, and simulate the honest parties sending shares on this polynomial to the adversary. When $|I| = t$ this is a deterministic process as we already have $t + 1$ shares that are determined on the output wires; however, when $|I| < t$, the simulator needs to generate some new shares on that output wires.

The assumption on the circuit. Damgård and Neilsen assumed that the output wires are direct outputs of multiplication gates. As a result, the shares on each output wire are independent of each other – and the simulator can just choose additional $|I| - t$ random shares on the output wire to interpolate the polynomial on that wire. When considering arbitrary circuits, output wires that are not a result of a multiplication gate may have some linear relation between them. To illustrate the challenge, consider output wires o_1 , o_2 , and o_3 , corresponding to three corrupted parties P_1 , P_2 , and P_3 , respectively, such that $o_3 = o_1 + o_2$. Denote the output values by y_1 , y_2 , and y_3 , respectively. The shares that the parties hold on the output wires define polynomials $g_1(x)$, $g_2(x)$ and $g_3(x)$, respectively. In the real execution it holds that $g_3(x) = g_1(x) + g_2(x)$, and therefore the simulator must choose the shares on the output wires wisely to guarantee the same dependency, which makes the simulation more challenging.

15:10 Static vs. Adaptive Security in Perfect MPC

Our static simulator. To guarantee such consistencies, our static simulator generates t random shares on each input wire and each output of a multiplication wire. That is, while the adversary corrupts some set I , the simulator, “in its head,” chooses an arbitrary set $\hat{I} \supseteq I$ of cardinality exactly t and simulates the shares for all parties in \hat{I} , while giving the adversary only shares for the parties in I . As a result, we have no random choice when simulating the output wires. The simulator holds t shares on each output wire, and also knows the constant term on the output wires of the corrupted parties. It can deterministically interpolate the polynomials on the output wire, and simulate the honest parties sending their shares on those wires. This guarantees that if there is any dependency between output wires, then the simulator provides consistent shares.

Our adaptive simulator. Assume that the adaptive adversary corrupts some party P_{i^*} . If $i^* \in \hat{I} \setminus I$, then providing the view of that party is easy. The simulator has already sampled all the shares that P_{i^*} is supposed to receive. On the other hand, if $i^* \notin \hat{I} \setminus I$, then we face a new obstacle. This is because the simulator has already defined t shares on each wire, and defining an additional share on each wire would completely determine each polynomial. Let $j^* \in \hat{I} \setminus I$. The key idea of our adaptive simulator is to “replace” the sampling of shares for party P_{j^*} with sampling shares for party P_{i^*} . Specifically, we show that each random choice made for P_{j^*} that leads to the current view of the adversary, can also be interpreted by a random choice made for P_{i^*} that leads to the exact same view. This procedure then changes the set of the simulator and “forgets” all the random choices made for P_{j^*} , but instead samples matching choices for P_{i^*} . This is essentially sampling random shares for P_{i^*} on the input wires of all honest parties and the outputs of f_{mult} , under the constraints posed by the shares of P_{j^*} on the output wires. Such sampling can be performed efficiently by solving a linear set of equations. Then, we are back to the previous case, where the corruption is made on some $i^* \in \hat{I} \setminus I$.

3 Preliminaries

Our results hold in any natural model that captures perfect adaptive security, for example, [10, 7, 16, 8, 25, 21]. For concreteness, we will state our results using the modular (non-concurrent) composability framework of Canetti [7]. Indeed, the separation in this limited setting extends to any of the models listed above, whereas our positive results translate to the universal-composability setting via the transformation in [24]. Before describing the security model, we give basic notation and define the cryptographic primitive used in our separation.

Notation. We denote by λ the security parameter. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. Let poly denote the set of all positive polynomials and let PPT denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\nu: \mathbb{N} \rightarrow [0, 1]$ is *negligible* if $\nu(\lambda) < 1/p(\lambda)$ for every $p \in \text{poly}$ and large enough λ . Given a random variable X , we write $x \leftarrow X$ to indicate that x is selected according to X , and given a set \mathcal{X} we write $x \leftarrow \mathcal{X}$ to indicate that x is selected uniformly at random from \mathcal{X} .

One-way permutations. Our separation in Section 4 will rely on the existence of a one-way permutation (OWP); that is a one-way function which is length preserving and one-to-one; we refer the reader to [19] for more details.

► **Definition 4 (OWP).** A polynomial-time function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is a one-way permutation if the following conditions are satisfied.

1. For every $\lambda \in \mathbb{N}$, f induces a permutation over $\{0,1\}^\lambda$, i.e., $f : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ is one-to-one and onto.
2. There exists a deterministic polynomial-time algorithm A such that on input $x \in \{0,1\}^*$ the algorithm A outputs $f(x)$.
3. For every PPT algorithm \mathcal{A} , every positive polynomial $p(\cdot)$, and all sufficiently large λ 's, it holds that

$$\Pr_{x \leftarrow \{0,1\}^\lambda} [\mathcal{A}(f(x)) = x] < \frac{1}{p(\lambda)}.$$

4 Static Security Does Not Imply Adaptive Security

In this section we prove Theorem 1. We show a functionality together with a protocol that privately computes it with perfect static security and efficient simulation. Further, we show that if the protocol privately computes the functionality with perfect adaptive security and efficient simulation, then one-way permutations do not exist.

We start by defining the functionality and the protocol. Next, in Lemma 6 we prove static security and in Lemma 9 we prove that adaptive security with efficient simulation cannot be achieved assuming OWP. Combined, this proves Theorem 1.

4.1 The Functionality

The n -party functionality f_{sum} is parametrized by a finite binary field \mathbb{F}_{2^ℓ} such that $2^\ell > n$. Looking ahead, having a binary field will enable using a one-way permutation $\text{OWP} : \{0,1\}^* \rightarrow \{0,1\}^*$ that is applied on a vector of field elements in a clean way. During the proof below, we will denote the field by $\mathbb{F} := \mathbb{F}_{2^\ell}$. The private input of every party is a polynomial of degree (at most) $n-1$ over \mathbb{F} , and the common output is the sum of the free coefficients.

- **Input:** The input of party P_i is a polynomial $g_i(x)$ of degree $n-1$, that is define by n coefficients $a_{i,0}, \dots, a_{i,n-1} \in \mathbb{F}$ such that $g_i(x) = \sum_{k=0}^{n-1} a_{i,k} x^k$.
- **Output:** The output of every party is $\sum_{i=1}^n g_i(0) = \sum_{i=1}^n a_{i,0}$.

4.2 The Protocol

The n -party protocol π_{sum} is parametrized by the field \mathbb{F} and assumes the existence of a one-way permutation $\text{OWP} : \mathbb{F}^{n-1} \rightarrow \mathbb{F}^{n-1}$, i.e., $\text{OWP} : \{0,1\}^{\ell(n-1)} \rightarrow \{0,1\}^{\ell(n-1)}$.

Protocol 5: Separating Protocol π_{sum} .

- **Private input:** The input of party P_i is a polynomial $g_i(x)$ of degree $n-1$ over \mathbb{F} .
- **Randomness:** The random tape of party P_i is $\rho_i \leftarrow \mathbb{F}^{n-1}$.
- **Common inputs:** n distinct non-zero elements $\alpha_1, \dots, \alpha_n \in \mathbb{F}$.
- **The protocol: code for party P_i :**
 1. Compute $(r_{i,1}, \dots, r_{i,n-1}) = \text{OWP}(\rho_i)$.
 2. Consider the polynomial

$$\begin{aligned} h_i(x) &:= g_i(x) + r_{i,1}x + \dots + r_{i,n-1}x^{n-1} \\ &= a_{i,0} + (a_{i,1} + r_{i,1})x + (a_{i,2} + r_{i,2})x^2 + \dots + (a_{i,n-1} + r_{i,n-1})x^{n-1}. \end{aligned}$$

15:12 Static vs. Adaptive Security in Perfect MPC

3. Send to every P_j its share $\gamma_{i \rightarrow j} := h_i(\alpha_j)$.
 4. Having received all shares $\gamma_{1 \rightarrow i}, \dots, \gamma_{n \rightarrow i}$, party P_i locally computes $\beta_i = \sum_{j=1}^n \gamma_{j \rightarrow i}$ and sends β_i to all other parties.
 5. Having received β_1, \dots, β_n , party P_i finds the unique degree- $(n-1)$ polynomial $H(x)$ satisfying $H(\alpha_j) = \beta_j$ for every $j \in [n]$.
- **Output:** $H(0)$.
-

4.3 Static Security

We start by proving static security of π_{sum} . We note that static security holds even if OWP is simply a permutation that is not necessarily one way and can be inverted efficiently.

► **Lemma 6.** *Protocol π_{sum} statically $(n-1)$ -privately computes f_{sum} with perfect semi-honest security and efficient simulation.*

Proof. Let \mathcal{A} be a static semi-honest adversary and let $I \subset [n]$ of size $|I| < n$ denote the set of corrupted parties' indices. We construct a simulator \mathcal{S} as follows:

1. The simulator initially receives auxiliary information z and the inputs of the corrupted parties $(g_i(x))_{i \in I}$. First, \mathcal{S} sends $(g_i(x))_{i \in I}$ to the trusted party and receives back the output value y . Next, \mathcal{S} invokes \mathcal{A} on z and $(g_i(x))_{i \in I}$.
2. To simulate the first round, for every $j \notin I$, the simulator chooses a random $\rho_j \leftarrow \mathbb{F}^{n-1}$ and computes $(r_{j,1}, \dots, r_{j,n-1}) = \text{OWP}(\rho_j)$. Next, \mathcal{S} chooses arbitrary polynomials $(\tilde{g}_j(x))_{j \notin I}$, each of degree at most $n-1$, under the constraint that

$$\sum_{j \notin I} \tilde{g}_j(0) = y - \sum_{i \in I} g_i(0),$$

and defines for every $j \notin I$

$$h_j(x) = \tilde{g}_j(x) + r_{j,1}x + \dots + r_{j,n-1}x^{n-1}.$$

Finally, for every $j \notin I$ and $i \in I$, the simulator sends $\gamma_{j \rightarrow i} := h_j(\alpha_i)$ to \mathcal{A} as the message from an honest P_j to a corrupted P_i , and receives the messages $\gamma_{i \rightarrow j}$ from \mathcal{A} as the message from a corrupted P_i to an honest P_j .

3. To simulate the second round, for every $j \notin I$, the simulator computes $\beta_j := \sum_{k=1}^n \gamma_{k \rightarrow j}$ and sends β_j as the message from P_j . Next, \mathcal{S} receives the message β_i from \mathcal{A} on behalf of every corrupted P_i .
4. Finally, \mathcal{S} outputs whatever \mathcal{A} outputs, and halts.

Note that by construction, the simulator \mathcal{S} runs in polynomial time in the running time of \mathcal{A} .

▷ **Claim 7.** $\{\text{REAL}_{\pi_{\text{sum}}, I, \mathcal{A}}(\mathbf{x}, z)\}_{(\mathbf{x}, z) \in (\{0,1\}^*)^{n+1}} \equiv \{\text{IDEAL}_{f_{\text{sum}}, I, \mathcal{S}}(\mathbf{x}, z)\}_{(\mathbf{x}, z) \in (\{0,1\}^*)^{n+1}}$.

Proof. Note that the only difference between the real execution of the protocol and the simulated execution in the ideal world is the construction of the constant terms of the polynomials $(h_j(x))_{j \notin I}$:

- In the real protocol, these constant terms are the constant terms of the original inputs of the honest parties $(g_j(x))_{j \notin I}$. In this case, by the linearity of Shamir's secret sharing and by the correctness of the interpolation, it holds that the output equals $\sum_{i=1}^n g_i(0)$.

- In the simulated execution, these constant terms are the constant terms of the polynomials $(\tilde{g}_j(x))_{j \notin I}$. These polynomials are generated under the constraint that $\sum_{j \notin I} \tilde{g}_j(0) = y - \sum_{i \in I} g_i(0)$, where y is computed by the trusted party to be $\sum_{i=1}^n g_i(0)$. It follows that

$$\sum_{j \notin I} \tilde{g}_j(0) = \sum_{j \notin I} g_j(0).$$

The claim now follows by the perfect privacy of Shamir's secret sharing. \triangleleft

This concludes the proof of Lemma 6. \blacktriangleleft

4.4 Inefficient Adaptive Security

By the construction of the static simulator it is clear that the simulation is straight-line and black-box; further, since we consider a semi-honest adversary that in particular does not change the corrupted parties' inputs, "round zero" (the beginning of the protocol) can be set as the committal round. Therefore, we can use [9] to derive the following corollary.

► **Corollary 8.** *Protocol π_{sum} adaptively $(n-1)$ -privately computes f_{sum} with perfect semi-honest security.*

4.5 No Efficient Adaptive Simulator

We proceed to prove that an efficient adaptive simulator can be used to invert the one-way permutation.

► **Lemma 9.** *Assume that OWP is a one-way permutation. Then, Protocol π_{sum} does not adaptively $(n-1)$ -privately compute f_{sum} with perfect semi-honest security and efficient simulation.*

Proof. Let λ denote the security parameter, i.e., the one-way permutation is defined as $\text{OWP} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$. For simplicity, assume that $\lambda = (n-1) \log(n+1)$ for some n such that $n+1$ is a power of 2 (this holds without loss of generality, since the security of the OWP holds for all sufficiently large λ 's). We consider the n -party functionality f_{sum} that is defined with respect to the field $\mathbb{F} = \mathbb{F}_{2^\ell}$ where $\ell = \log(n+1)$; then, indeed, $|\mathbb{F}| > n$ as required and OWP induces a permutation over \mathbb{F}^{n-1} .

Defining the adversary and the environment. Consider the following adaptive, semi-honest, $(n-1)$ -limited adversary \mathcal{A} and the environment \mathcal{Z} for π_{sum} .

1. The environment does not send any auxiliary information to the adversary at the beginning.
2. The adversary \mathcal{A} starts by corrupting the parties P_3, \dots, P_n and learns their inputs $(g_3(x), \dots, g_n(x))$ and random tapes (ρ_3, \dots, ρ_n) . The environment does not send any auxiliary information to the adversary for these corruptions.
3. Next, \mathcal{A} receives first-round messages $\gamma_{1 \rightarrow 3}, \dots, \gamma_{1 \rightarrow n}$ from P_1 and $\gamma_{2 \rightarrow 3}, \dots, \gamma_{2 \rightarrow n}$ from P_2 .
4. The adversary completes the execution honestly and outputs its view.
5. The environment corrupts P_1 in the PEC phase and learns (amongst other things) its input $g_1(x)$, randomness ρ_1 , and the values $(r_{1,1}, \dots, r_{1,n-1})$ computed in Step 1 of π_{sum} .
6. The environment checks whether $(r_{1,1}, \dots, r_{1,n-1}) = \text{OWP}(\rho_1)$; if so it outputs real and otherwise ideal.

15:14 Static vs. Adaptive Security in Perfect MPC

The corresponding adaptive simulator. By the assumed security of π_{sum} , there exists an efficient adaptive simulator \mathcal{S} for \mathcal{A} and \mathcal{Z} . Note that by construction, the adversary \mathcal{A} runs in polynomial time with respect to the parameter λ ; hence, \mathcal{S} is PPT with respect to λ . We will use \mathcal{S} to construct a PPT inverter \mathcal{D} for the OWP.

Constructing the inverter from the simulator. The inverter \mathcal{D} receives as input the challenge $y^* \in \{0, 1\}^\lambda$. We will consider y^* as element of \mathbb{F}^{n-1} , i.e., $y^* = (r_1^*, \dots, r_{n-1}^*) \in \mathbb{F}^{n-1}$. The inverter \mathcal{D} proceeds as follows:

1. \mathcal{D} invokes the simulator \mathcal{S} and emulates the ideal computation of f_{sum} toward \mathcal{S} ; initially, \mathcal{D} sends nothing as the auxiliary information to \mathcal{S} .
2. When \mathcal{S} corrupts the parties P_3, \dots, P_n in the emulated ideal computation, \mathcal{D} chooses arbitrary polynomials $(g_3(x), \dots, g_n(x))$ of degree at most $n-1$ over \mathbb{F} and hands $g_i(x)$ to \mathcal{S} as the input value of P_i ; \mathcal{D} sends nothing as the auxiliary information to \mathcal{S} .
3. When \mathcal{S} sends inputs to the trusted party on behalf of the corrupted parties, \mathcal{D} responds with an arbitrary output value $y \in \mathbb{F}$.
4. Once \mathcal{S} generates its output, containing the view of \mathcal{A} , the inverter \mathcal{D} extracts the first-round messages that each corrupted party received from the honest party P_1 , denoted $\gamma_{1 \rightarrow 3}, \dots, \gamma_{1 \rightarrow n}$.
5. \mathcal{D} samples two field elements $\gamma_1, \gamma_2 \leftarrow \mathbb{F}$ and interpolates the unique polynomial h of degree $n-1$ satisfying the constraints:

$$h(\alpha_1) = \gamma_1, \quad h(\alpha_2) = \gamma_2, \quad h(\alpha_i) = \gamma_{1 \rightarrow i} \text{ for } i \in \{3, \dots, n\}.$$

Next, \mathcal{D} computes $g_1(x) = h(x) - \sum_{k=1}^{n-1} r_k^* x^k$.

6. \mathcal{D} sends a “corrupt P_1 ” request to \mathcal{S} during the PEC phase. When \mathcal{S} corrupts P_1 in the emulated ideal computation, \mathcal{D} sets the input of P_1 to be $g_1(x)$. Next, \mathcal{S} responds with the view of P_1 , containing the content of its random tape ρ_1 .
7. \mathcal{D} outputs ρ_1 .

Efficiently inverting the OWP. First, notice that by construction, the running time of the inverter \mathcal{D} is polynomial with respect to its input y^* and to the running time of \mathcal{S} ; therefore, \mathcal{D} is PPT with respect to the parameter λ . We proceed to show that the success probability of \mathcal{D} in inverting a random challenge $y^* \leftarrow \{0, 1\}^\lambda$ is non-negligible.

▷ **Claim 10.** $\Pr_{y^* \leftarrow \mathbb{F}^{n-1}} \left[\text{OWP}(\mathcal{D}(y^*)) = y^* \right] = \frac{1}{|\mathbb{F}|^2}$.

Proof. In our proof, we do not assume any specific *behavior* of the adaptive simulator \mathcal{S} (i.e., we do not know *how* the simulator generates its output messages). However, based on the assumed perfect security of the protocol, the adaptive simulator \mathcal{S} operates according to the following interface:

1. \mathcal{S} corrupts parties P_3, \dots, P_n and learns their inputs $(g_3(x), \dots, g_n(x))$.
2. \mathcal{S} sends the inputs $(g_3(x), \dots, g_n(x))$ to the trusted party and obtains the output value y .
3. \mathcal{S} generates the simulated view of the adversary, which in particular contains the messages $\gamma_{1 \rightarrow 3}, \dots, \gamma_{1 \rightarrow n}$ from P_1 to parties P_3, \dots, P_n .
4. \mathcal{S} corrupts party P_1 and learns its input $g_1(x)$.
5. \mathcal{S} outputs the view of P_1 , including its random tape ρ_1 and values $(r_{1,1}, \dots, r_{1,n-1}) = \text{OWP}(\rho_1)$, such that the polynomial $h_1(x) = g_1(x) + \sum_{k=1}^{n-1} r_{1,k} x^k$ satisfies $h_1(\alpha_i) = \gamma_{1 \rightarrow i}$ for $i \in \{3, \dots, n\}$.

Recall that in Step 5 of the construction of the inverter, \mathcal{D} samples $\gamma_1, \gamma_2 \leftarrow \mathbb{F}$ and interpolates the polynomial $h(x)$ that satisfies the following constraints:

$$h(\alpha_1) = \gamma_1, \quad h(\alpha_2) = \gamma_2, \quad h(\alpha_i) = \gamma_{1 \rightarrow i} \text{ for } i \in \{3, \dots, n\}.$$

First, for every choice of $\gamma_1, \gamma_2 \in \mathbb{F}$ there exists a unique polynomial of degree $n - 1$ over \mathbb{F} satisfying these constraints. In the other direction, every polynomial $h'(x)$ of degree at most $n - 1$ over \mathbb{F} , satisfying $h'(\alpha_i) = \gamma_{1 \rightarrow i}$ for $i \in \{3, \dots, n\}$, induces two points $\gamma_1 = h'(\alpha_1)$ and $\gamma_2 = h'(\alpha_2)$ in \mathbb{F} .

It follows that the inverter succeeds in inverting y^* if and only if $h_1(\alpha_1) = \gamma_1$ and $h_1(\alpha_2) = \gamma_2$ (where $h_1(x)$ is the polynomial defined by \mathcal{S}). Indeed, in this case $h_1(x) = h(x)$, which means that $h_1(x) - g_1(x) = h(x) - g_1(x)$, i.e., $\sum_{k=1}^{n-1} r_{1,k} x^k = \sum_{k=1}^{n-1} r_k^* x^k$, and finally $(r_{1,1}, \dots, r_{1,n-1}) = (r_1^*, \dots, r_{n-1}^*)$. Since γ_1 and γ_2 are sampled uniformly at random from \mathbb{F} , this happens with probability $1/|\mathbb{F}|^2$. \triangleleft

Finally, note that $y^* \in \mathbb{F}^{n-1}$, i.e., $y^* \in \{0, 1\}^{(n-1)\log(n+1)} = \{0, 1\}^\lambda$. However, the inverting probability is $1/|\mathbb{F}|^2 = 1/2^{2\log(n+1)} = 1/(n+1)^2$, which is non-negligible in λ . We conclude that \mathcal{D} is PPT in λ and succeeds in inverting OWP with non-negligible probability. This contradicts the assumption that OWP is a one-way permutation. \blacktriangleleft

References

- 1 Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly-secure multiparty computation. IACR Cryptol. ePrint Arch., 2011. URL: <http://eprint.iacr.org/2011/136>.
- 2 Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017.
- 3 Michael Backes, Jörn Müller-Quade, and Dominique Unruh. On the necessity of rewinding in secure multiparty computation. In *Proceedings of the Fourth Theory of Cryptography Conference, TCC 2007*, pages 157–173, 2007.
- 4 Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology – EUROCRYPT 1992*, pages 307–323, 1992.
- 5 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.
- 6 Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. Must the communication graph of MPC protocols be an expander? In *Advances in Cryptology – CRYPTO 2018, part III*, pages 243–272, 2018.
- 7 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
- 8 Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
- 9 Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, 2004.
- 10 Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648, 1996.
- 11 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

15:16 Static vs. Adaptive Security in Perfect MPC

- 12 Ran Cohen, Juan A. Garay, and Vassilis Zikas. Completeness theorems for adaptively secure broadcast. IACR Cryptol. ePrint Arch., 2021. URL: <http://eprint.iacr.org/2021/775>.
- 13 Ran Cohen, Abhi Shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In *Advances in Cryptology – CRYPTO 2019, part II*, pages 30–60, 2019.
- 14 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology – EUROCRYPT 1999*, pages 311–326, 1999.
- 15 Ivan Damgård and Jesper Buus Nielsen. Adaptive versus static security in the UC model. In *Proceedings of the 8th International Conference on Provable Security (ProvSec)*, Lecture Notes in Computer Science, pages 10–28, 2014.
- 16 Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In *Advances in Cryptology – CRYPTO 2000*, pages 74–92, 2000.
- 17 Juan A. Garay, Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. The price of low communication in secure multi-party computation. In *Advances in Cryptology – CRYPTO 2017, part I*, pages 420–446, 2017.
- 18 Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In *Advances in Cryptology – CRYPTO 2012*, pages 105–123, 2012.
- 19 Oded Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.
- 20 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.
- 21 Martin Hirt, Chen-Da Liu-Zhang, and Ueli Maurer. Adaptive security of multi-party protocols, revisited. In *Proceedings of the 19th Theory of Cryptography Conference, TCC 2021, part I*, pages 686–716, 2021.
- 22 Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In *Advances in Cryptology – EUROCRYPT 2010*, pages 466–485, 2010.
- 23 Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In *Proceedings of the 16th International Conference on the Theory and Practice of Public-Key Cryptography (PKC)*, pages 14–31, 2013.
- 24 Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39(5):2090–2112, 2010.
- 25 Ralf Küsters, Max Tuengerthal, and Daniel Rausch. The IITM model: A simple and expressive model for universal composability. *Journal of Cryptology*, 33(4):1461–1584, 2020.
- 26 Huijia Lin, Tianren Liu, and Hoeteck Wee. Information-theoretic 2-round MPC without round collapsing: Adaptive security, and more. In *Proceedings of the 18th Theory of Cryptography Conference, TCC 2020, part II*, pages 502–531, 2020.
- 27 Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *Journal of Cryptology*, 24(4):761–799, 2011.
- 28 Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *Advances in Cryptology – CRYPTO 1991*, pages 392–404, 1991.
- 29 Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO 2002*, pages 111–126, 2002.
- 30 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.
- 31 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.