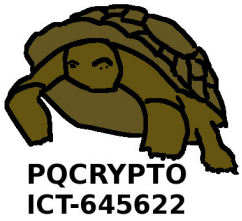

PQCRYPTO

Post-Quantum Cryptography for Long-Term Security

Project number: Horizon 2020 ICT-645622

D3.5

D3.5 Cloud: Advanced applications

Due date of deliverable: February, 28th 2018
Actual submission date: April 13, 2018

WP contributing to the deliverable: WP3

Start date of project: 1. March 2015 Duration: 3 years

Coordinator:
Technische Universiteit Eindhoven
Email: coordinator@pqcrypto.eu.org
www.pqcrypto.eu.org

Revision 0.001

Project co-funded by the European Commission within Horizon 2020		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

D3.5 Cloud: Advanced applications

Daniel Augot, André Chailloux, Matthieu Rambaud

April 13, 2018
Revision 0.001

The work described in this report has in part been supported by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Abstract

This document recommends to use *secure multiparty computation* as a conclusion of the Task D3.3 *Advanced applications* of the work package *WP3: Post-quantum cryptography for the cloud*. Typically, advanced applications involve both encryption and computing: this is for example the case when a user delegates her data to a remote system, in an encrypted form. Yet, the user, the system, or a third party may want to extract (autorized) results of some computation on the secret data, without leaking no more information than the result of the computation. Secure multiparty computation enables exactly to do this when instantiated in a proper way.

We report here on the techniques of multiparty computation and the security models for which the protocols are proven secure. Multiparty computation has several strong advantages relevant for post-quantum cryptography: it does not rely on a particular hardness assumption and is information theoretically secure. This implies that whatever the adversary, even with infinite or quantum power, all leaked information has no correlation with the hidden data: this is the same exact security as the one-time pad which is well known to be information theoretically secure. Furthermore, secure multiparty computation is fast and huge progress has been achieved recently on the efficiency issues, making it more than 100 times faster than homomorphic encryption ([EMV17]

Keywords: multi-party computation, secret sharing, post-quantum, cloud computing

Contents

1 Summary **2**
1.1 Encryption and computing, FHE versus MPC . . . 2
1.2 Organisation of the document . . . 3

2 Definitions **4**
2.1 Basic definitions . . . 4
2.1.1 Generalities . . . 4
2.2 Security definition in the UC framework . . . 7

3 *Perfect security* when less than a *third* of participants cheat. **8**
3.1 Summary of the section . . . 8
3.1.1 Theory . . . 8
3.1.2 Efficiency issues . . . 9
3.2 Perfect security with $t < n/2$ semi-honest adversaries . . . 10
3.3 Perfect security with $t < n/3$ adversaries . . . 10
3.3.1 Preliminary: need to implement a broadcast channel for $t < n/3$. . . . 10
3.3.2 Principle: upgrading the protocol with perfectly secure commitments . . 12
3.3.3 Realizing a perfectly secure commitment protocol, and why the limit $n/3$ 14

4 *Statistical security* when less than *half* of participants cheat, *conditionned to broadcast or correlated randomness* **17**
4.1 Summary of the section . . . 17
4.1.1 Theory . . . 17
4.1.2 Efficiency issues . . . 17
4.2 How assuming access to a broadcast channel enables the bound $t < n/2$ . . . 18
4.2.1 Main idea: signing each message during the commitment protocol . . . 18
4.2.2 Implementing IC signatures . . . 18
4.3 Assuming access to a source of correlated randomness also enables the bound $t < n/2$ . . . 19

5 Breaking the honest majority limit with more crypto, and the problem of early aborters **21**
5.1 Summary of the section . . . 21
5.2 Preprocessing for a fast and secure online phase . . . 22
5.2.1 Principle (the semi-honest case) . . . 22
5.2.2 Protocols robust against $t < n$ active adversaries, with possibility of abort 22
5.3 Identifiable abort . . . 24
5.4 Security against *covert adversaries*: ones that do not want being seen cheating or aborting . . . 24
5.4.1 Security notion . . . 24
5.4.2 State of the art for covert security . . . 24
5.5 Public auditability . . . 25
5.5.1 Public blame of cheaters . . . 25
5.5.2 Public proof of correctness of the result . . . 26
5.6 When synchronicity or completeness of the network cannot be achieved . . . 26

5.6.1 Asynchronous MPC 26
5.6.2 Server-based one-pass protocols 26
5.6.3 Non-interactive MPC 27

6 Versatility of use-cases and best general protocols 27
6.1 MPC and encryption (AES) 27
6.2 MPC as a computing network 28
6.3 MPC as a privacy preserving reporting tool 28
6.4 Best available generalistic protocols 28
6.4.1 Are specific protocols faster ? 28
6.4.2 Fastest generalistic implemented protocols 29

1 Summary

1.1 Encryption and computing, FHE versus MPC

PQCrypto H2020 projet is devoted to post-quantum, long term security, since in particular, it is aimed at preparing for a post-quantum future, when most cryptographic primitives nowadays used will be “broken”, may this happening fifty years from now. Most objectives of the projet are devoted to classical and fundamental cryptographic primitives, encryption (symmmetric and asymmetric), public-key signatures, etc.

Work package 3 is devoted to the particular setting of applications for the cloud, meaning interactions between a particular user or simple system with a remote and powerful system, managed by a single corporation, like Google Drive, OVH, Dropbox. Workpackage 3 proposes several basic subproblems, like **3.1 encrypt at home**, **3.2 share files**, and finally **3.3 advanced applications**, which is the object the current document.

Advanced applications typically involve both computation and encryption, where some output (encrypted or not) is obtained as a result of a computation on encrypted or secret data. The two principal approaches to this broadly defined area are so-called *homomorphic encryption* and *multi-party computation*. (Fully) homomorphic encryption (FHE) enables to perform computations on several ciphertexts with associated plaintexts, to obtain a new ciphertext, which is such that, when decrypted, the obtain plaintext is the same as the result of these computations on the associated plaintexts. It can be seen a classical public-key encryption scheme, with one more features which is homomorphic computation, and as such, relies on the classical paradigm of a trapdoored hard problem. Typically, lattice based cryptography forms the ground for FHE. On the other hand, secure multiparty computation (SMC, also known as multi-party computation, MPC), allows two or more parties to perform a computation over their inputs while hiding the inputs from each other. MPC has furthermore the property that it can be made information theoretically secure (IT-secure), which means that it does not rely on some hardness assumption. So security holds against adversaries with unlimited computing power, be it classical or quantum computing. This property has furthermore a positive consequence on performance: citing the master’s thesis [Zys06] (linked with MIT’s Enigma) "As information theoretically secure-MPC protocols do not depend on the hardness of specific computational problems, they often admit faster implementations using a smaller prime field for data representations ([DGH12] , [ELC12], [WLC14])".

Beyond the application of MPC for the cloud as a privacy preserving computing platform, the versatility of these protocols allow to address any problem where no trusted third party

exists. The most known use-cases are auctions and coercion-proof voting [DKR06, CCM08], or the less known *matchmarking* problem [CDN, p 10]. Consider a set of companies, each of whom having a private list of companies that it would like to do business with. The goal of the computation is to match companies that mutually want to do business with, without revealing the private lists. Popular variants today are (byzantine) consensus algorithms for blockchains [CGLR17]. The most currently needed functionality might be public secure random generation, which is not a simple problem —as illustrated with the Dual EC DRBG backdoor [Fer07, BLN16]—, for cryptography or blockchain proof of stake applications: see [CD17, KRDO17, SJK+17]. Finally, MPC makes possible to perform cross-mining and numerical analysis in sensible databases (police, bank, social network users) without downloading nor merging them: see [HHIL+17, UftFPotE15, ZNP15].

With PQCrypto objectives in mind (security within 50 years), and that lattice-based cryptography is relatively new (meaning that cryptanalysis of it has not reach a sufficient level of knowledge to set the size of the keys for long term security), we have decided to restrict the discussion to MPC. Furthermore, MPC is becoming more and more efficent, orders of magnitude faster than FHE [EMV17].

1.2 Organisation of the document

Research in the MPC area has been ongoing for more than thirty years, and this document can not thoroughly review all the research in this area (an heavy book would not even suffice). We will essentially focus on most important and relevant results, while at the same time giving a flavor of the techiques used in MPC, mainly *secret sharing* and cooperation between players to obtain sub-primitives, like *broadcast* or *homomorphic commitments*.

We have tried to make the exposition simple, well-organized and straigthforward, with the (admitedly questionnable) choice of leaving sub-topics, minor improvements, disgression, etc, in footnotes. Indeed, over 35 years of active research, MPC has grown into a rich and complex topic, with many incomparable flavors and numerous protocols and techniques [IKP+16]: just cataloguing the state of the art results is a non-trivial research project in itself, as exemplified by the recent work of Perry et al. [PGFW14], which proposes classifying the existing protocols using 22 dimensions. Similarly, we do not go into full length formal presentations and definitions of the security models and theorems, which would need a lot of space and notation, with the drawback of driving the reader away from core ideas. Thus we given informal yet hopefully sensible presentations of the definitions, protocols and security results, to allow the reader to get a flavor of the area.

The review is organized as follows. In Section 2, we review (informal) basic definitions, and the security model we will restrict to and which is the *strongest possible security requirement: Canetti's universal composability* (UC). It guarantees that a UC protocol will remain secure when arbitrarily composed or executed in parallel with other protocols. Thus we less discuss GMW-like protocols or Yao's/BMR garbled circuits techniques [Yao82, BMR90].

In Section 3, we review the protocols which are *unconditionally secure, even in the quantum setting*, provided that less than one third of participants cheat. The limit is raised up to half of participants, if cheaters are just honest-but-curious.

In Section 4, we review protocols which are also quantum secure, but with a less stringent security goal than perfect security: *stastitical security*, where the adversary, interacting with a run of a MPC protocol, has a non-zero but negligible probability of getting some information. This is doable within the limit of one-half corrupted players.

Then, in Section 5, we review even less stronger security, yet quite relevant, security models, which allow to break the one half limit, and to have a correct execution of the protocol, even when all the players but one are corrupted. These security models essentially allow the protocol to *abort* or fail due to some corrupted players, yet the honest players will learn who had such a corrupt influence on the protocol. This knowledge can be made auditable and transferable to an external party who did not participate in the protocol, like a judge.

Finally, Section 6, we present how MPC protocols can be versatilely adaped to several use cases, which are not exactly the one given by the definition, like MPC for auditability, and MPC as privacy preserving computing platform.

2 Definitions

2.1 Basic definitions

2.1.1 Generalities

First we summarize the strongest possible security notion, called '"Universal composability", following the seminal article of [Can01] and borrowing freely from [CDN, chap 4], but with a simplified presentation.

Players $P_1, \ldots, P_n$ hold secret *input values* $x_1, \ldots x_n$ (e.g. their ages), and want to obtain a *common result*[1] $y = f(x_1, \ldots, x_n)$ from them (e.g. the median age), where f is previously agreed upon function. An ideal *functionality* $\mathcal{F}_f$ represents what we expect from secure multiparty computation: $\mathcal{F}_f$ is an imaginary machine, with which each player P_i interacts individually and privately (e.g. P_i gives $\mathcal{F}_f$ his secret x_i, without leaking it to other players)[2] . Then, $\mathcal{F}$ returns the result y to every $P_1, \ldots, P_n$ then blows up, leaving no trace of secrets $(x_i)_i$, neither any data about its internal states, if any, see Fig. 2.1.1.

Note than the ideal functionnality can actually leak information, depending on the very nature of the function f. Consider for instance P_1, P_2, P_3 three players who about to ask $\mathcal{F}_{median}$ for their median age. But suppose that the real agenda of malicious players P_1 and P_2 is to learn the age of P_3. Then P_1 and P_2 are going to give false inputs to $\mathcal{F}_f$: say $x_1 = 1$ and $x_2 = 200$. Thus $x_1 \leq x_3 \leq x_2$, so $\mathcal{F}_{median}$ will return the median input to everybody, which is x_3, the age of P_3. This example stresses that *it is not possible to prevent players from lying on their inputs, and to obtain some information about inputs of honest players from the final result.*

Adversarial (or malicious) players $P_1, \ldots, P_t$ are a subgroup of players who collude (w.l.o.g. we assume that they are the players $P_1, \ldots, P_t$, yet they are unknown to honest players). Roughly speaking, they want to discover secret inputs of other players. Then, as the example above illustrates, they can obtain information, depending on the function f, by tweaking the inputs they give to the ideal functionality: This is called *allowed influence* on the process. Thus, when implementing $\mathcal{F}_f$ by a real-life protocol π_f, what one can *expect at best* is that adversary players cannot gather more information or have more influence on the final result than in the ideal setting:

[1] We stress that this is equivalent to [Can01]. There, *environment* $\mathcal{E}$ learns basically: adversary $\mathcal{A}$'s inputs, the final result (communicated by $\mathcal{A}$), and public messages from honest players of type "success" or "abort".

[2] Canetti [Can01] allows *reactivity*: $\mathcal{F}$ can have a state that depends on previous interactions.

Definition 2.1 (Simplified security definition)**.** *We want that adversaries playing the protocol π_f have no more influence on the result (robustness)*[3]*; and gather no more information (privacy), than what they could be obtain from merely lying on their inputs (allowed influence on the ideal functionality)*[4] $\mathcal{F}_f$*.*

Definition 2.2. *Let $P_1, \ldots, P_n$ be a set of players. Suppose that (i) there are* secure authenticated point to point channels *between each pair of players P_i and P_j; (ii) and that players are able to communicate by* rounds of simultaneous communication*, during which each player P_i sends simultaneously messages $\{m_{i,j}\}_{i,j}$ to other players $(P_j)_j$, and this for all players P_i at the same time.* [5]

A protocol *π ([CDN, §1.3]) is then a set of instructions that players are supposed to follow to obtain the desired result.*

Typically these instructions consist in *when* a player P_i must send a message to P_j, or what must be the *format* of each message[6] m, and its content: see fig 2.1.1.

Definition 2.3. Honest players $P_{t+1}, \ldots, P_n$ *always follow the protocol.*

Semi-honest adversaries *(aka.* passive *or* honest but curious*): $P_1, \ldots, P_t$ always follow the instructions but share all their informations: they have a collective* view *of the protocol*[7]*.*

Active adversaries [CDN, §4.2.3] *(aka.* actively corrupted *or* malicious*) moreover don't necessarily follow instructions of the protocol.*

Definition 2.4. *In some cases, an ideal functionality $\mathcal{F}$ or a protocol π can be aborted by certain players, and this event is called* abort *. To simplify, we will consider only situations where all honest players are simultaneously informed that the functionality or protocol did abort*[8]*.*

Definition 2.5 (Perfect and statistical (IT) security - simplified)**.** *Perfect security: if Definition 2.1 always holds against an unbounded adversary, then we say that protocol $\pi_{\mathcal{F}}$* perfectly implements *$\mathcal{F}$. Statistical security: if perfect security always holds except in some executions of protocols where security is failed, then we say that protocol $\pi_{\mathcal{F}}$* statistically implements *$\mathcal{F}$. The probability of failure is assumed to be exponentially small in a security parameter k*[9] *(e.g. with a finite field of size 2^{128}, the probability of a security failure is 2^{-128})*

[3] In particular the protocol *always returns a result ("guaranteed output")* if the functionality $\mathcal{F}$ is meant to do so

[4] Simplifying [Can01], we do not mention other information gathered by *environment* $\mathcal{E}$. Corruption is always static, and finally we consider $\mathcal{E}$ and $\mathcal{A}$ as acting as a single entity (which is harmless, as noticed in [Can01, top of p4]).

[5] More formally we have assumed a synchronous environment with access to F_{PPC}: see [CDN, §4.3.5 & th 5.9]. Notice that malicious players (see below) can read the messages intended to them first, and then change their mind: [CDN, 4.4.2]

[6] A player not following the two previous instructions is likely to publicly provoque an abort . Robust protocols will typically have him excluded: see end of §3.3.2.

[7] Thus they are often seen as a single entity in the litterature: the *adversary* $\mathcal{A}$ (often forgetting environment $\mathcal{E}$, as we do here).

[8] In general it can be more complex: as stressed in [FGMvR02], an adversary player could well achieve that some players end the protocol thinking it ran successfully (e.g. for signature protocol: that a message was successfully signed although it was not).

[9] Notice the weaker notion of *partial fairness* where the probability of such cases is inverse polynomial, which is greater than negligible: see [IOS12, 5.3] for results in this case.

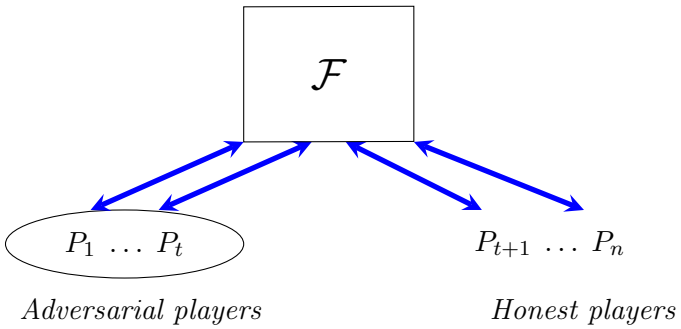

Figure 2.1: Ideal functionnality

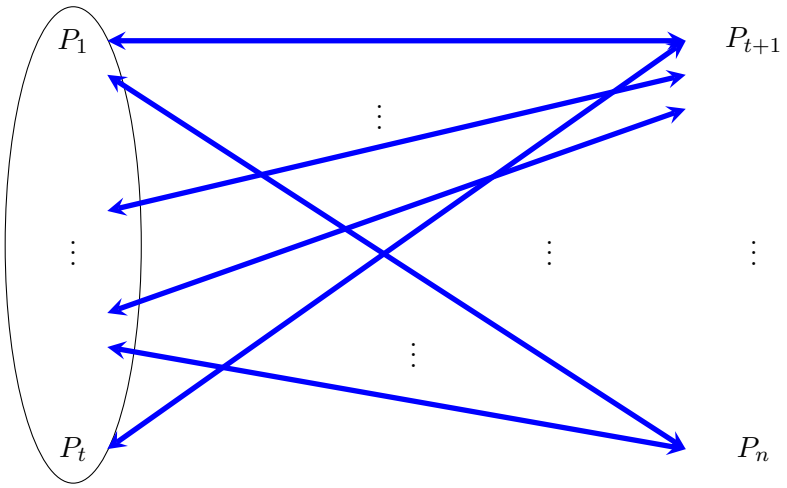

Figure 2.2: Real-life MPC protocol

We stress that these two security levels are *information-theoretically secure*, since the adversary is assumed unbounded.

Remark 2.1. Typically in the statistical case, the adversary tries to cheat and guess a random secret held by an honest player: if he guesses right, he will achieve a forbidden influence on the process[10]. In this unlucky event, perfect security does not then hold anymore. But if he fails, then the protocol abort s. *This probability of a correct guess decreases exponentially* in the bit-size of the secret, whatever computational power the adversary has.

2.2 Security definition in the UC framework

To say that adversaries have only an allowed influence means that, whatever they do during the protocol, everything happens as if they interacting with the functionnality $\mathcal{F}$. This formalized by:

Definitions 2.6 (Perfect and statistical (IT) security[11])**.** *We say that protocol $\pi_{\mathcal{F}}$* unconditionally perfectly (resp. statistically) implements a funcionnality $\mathcal{F}$ *if there exists to craft a* polytime[12] *interface $\mathcal{S}_{\mathcal{F}}$ between adversaries $P_1, \ldots, P_t$ and $\mathcal{F}$, the "simulator"*[13]*, such that adversaries with an illimited computational power*[14] *have no distinguishing advantage (resp. have a statistically small distinguishing advatange*[15] *) for making a difference between an actual execution of protocol $\pi_{\mathcal{F}}$ and the following situation, shown in Fig. 2.2, where:*

- *honest players $P_{t+1}, \ldots, P_n$ interact directly with $\mathcal{F}$*
- *the simulator $\mathcal{S}_{\mathcal{F}}$ interacts with the dishonest players $P_1, \ldots, P_t$, by impersonating honest players $P_{t+1,virtual}$, $\ldots$, $P_{n,virtual}$;*
- *the simulator $\mathcal{S}_{\mathcal{F}}$ interacts with $\mathcal{F}$ by impersonating the adversarial players $P_{t+1,virtual}$, $\ldots$, $P_{n,virtual}$.*

Here $\mathcal{S}_{\mathcal{F}}$ has no choice but to have an allowed influence.

We insist that $\mathcal{S}_{\mathcal{F}}$ only receives information from the output of $\mathcal{F}$: he doesn't learn the honest players' messages in the protocol, let alone their secret inputs. This guarantees that adversaries can't obtain more information than what is available from the ideal functionality (privacy: cf [CDN, §4.1.1 & §4.1.3]). Adversaries still hear final messages of type "success" or "abort" from the honest players (see footnote 1). So $\mathcal{S}$ should also act consistently with the success or failure of the protocol (see[CDN, p, 108]).

Remark 2.2. So the simulator has to work in two directions[16]:

[10] See e.g. [CDN, p109] or [DKL+13]: here, the cheater can try to guess a "MAC" and thus disguise a message of his own into an authenticated one from some other player.

[11] See also [FLNW17] and [DBP14, Th 3] for easier criterions of universal composability in particular cases.

[12] See [CDN, def 2.9 & 4.2.4]

[13] In the original UC model, and [CDN], where corrupt entities and the environment are disjoint entities, the simulator impersonates the adversary to the environment, but doesn't act as the environment tell him to (see also [Can06, figure 1]). As [Can01] puts it: "$\mathcal{S}$ now has to interacts with $\mathcal{E}$, throughout the execution, just as $\mathcal{A}$ did. Furthermore, $\mathcal{S}$ cannot "rewind" $\mathcal{E}$. Indeed, it is this pattern of free interaction between $\mathcal{E}$ and $\mathcal{A}$ that allows proving that security is preserved under universal composition.

[14] See [CDN, 4.3.4]

[15] [CDN, 2.2.2 & 2.3.1 & 2.3.2] fot the definitions of (statistical) indistinguishability. Here, κ should be understood as a security parameter, typically the input's size (size of the base field $\mathbf{F}$ in bits).

[16] In the [CDN] the simulator is *efficient*: he can't solve cryptographic problems. But he actually could in the proofs of security below, this wouldn't make any difference

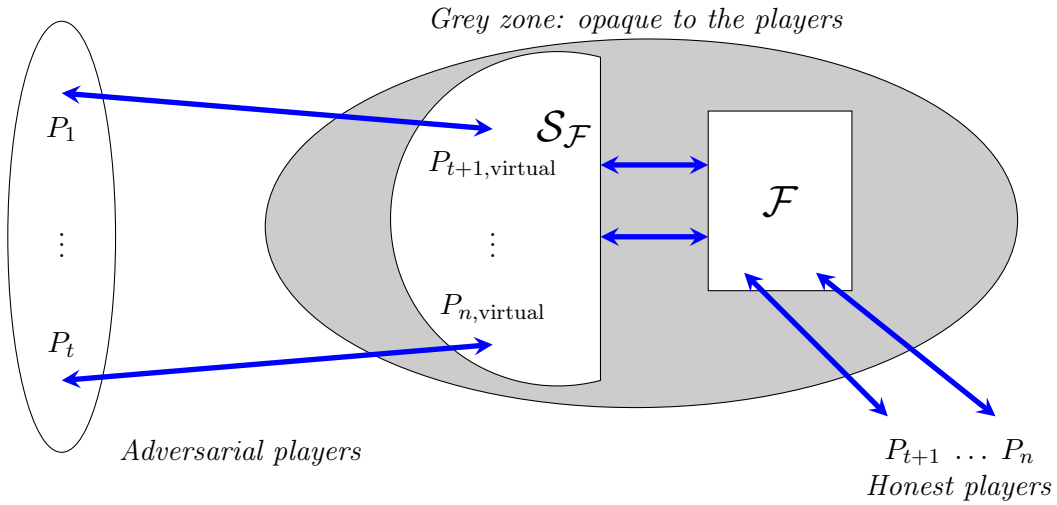

Figure 2.3: Equivalent view: a simulator interface for adversaries $P_1, \ldots, P_t$

⇒ It transforms the messages of (possibly cheating) adversaries playing the protocol into an allowed influence producing the same result (*robustness*: cf [CDN, p54]).

⇐ It returns messages to adversaries, so that they can't make the difference with a real protocol (*privacy*). The simulator has in fact only access to the output of the ideal functionality.

UC framework owns its success from Canetti's *universal composition theorem* [CDN, 4.20]. It states that, if π_f perfectly implements $\mathcal{F}_f$, then π_f can be used as a black box for $\mathcal{F}_f$ in any larger UC secure protocol without weakening its security.

Of even greater importance to us is Unruh's theorem [Unr10, th 2], that states that a UC secure protocol in the classic world remains UC secure in the quantum setting[17].

As stressed by [FLNW17], UC property also enables to run the protocol and subprotocols in *parallel* and *concurrently*, which enables to obtain a more general computation model, e.g. by running many executions in parallel or running layers of an evaluation circuit in parallel.

3 *Perfect security* when less than a *third* of participants cheat.

3.1 Summary of the section

3.1.1 Theory

Theorem 3.1 ([CDN, Th 3.6 & §3.4 ; Th 5.10 & Th 5.11])**.** *Any MPC functionality can be unconditionaly implemented with perfect security against* $t < n/2$ passive *adversaries. This bound is optimal. Any MPC functionality can be unconditionaly implemented with perfect UC security against* $t < n/3$ active *adversaries. This bound is optimal.*

Recall that the perfect security property is *information-theoretically secure*, meaning that the implementation protects against adversaries with unlimited computational power, for example quantum computers. It dates back to Ben-Or, Goldwasser and Wigderson [MBOW88]

[17] "That is, the protocol parties, the adversary, the simulator, and the environment are allowed to store, send, and compute with quantum state"

and independantly Chaum, Damgard and Crépeau [CCD88]. A possibly simpler approach can be found in [IKP+16]. Let us first study the simplest case, which is security against semi-honest (or passive) adversaries.

Let us precise that from now on, we will only consider circuits based on addition and multiplication in a finite field $\mathbf{F}$, as described by the ideal functionality $\mathcal{F}_{\text{SFE}}^{f}$ in Figure 3.1. There, the circuit performs the computation of $f(x_1, \ldots, x_n)$ on the inputs $x_i \in \mathbf{F}$, where f is a function composed of additions and multiplications (we simplify [CDN, p88]). Indeed, *any boolean circuit can be implemented with these operations* by [CDN, exercice 3.1].

$\mathcal{F}_{\text{SFE}}^{f}$

Input Every player P_i gives her input x_i to $\mathcal{F}_{\text{SFE}}^{f}$. Furthermore, adversarial players $P_1, \ldots P_t$ can change their minds on their inputs at anytime before the computation.

Computation and leak to the adversaries When all players ask $\mathcal{F}_{\text{SFE}}^{f}$ to do so, then it computes $f(x_1, \ldots, x_n)$ and delivers this in advance to adversarial players.

Output When all players ask $\mathcal{F}_{\text{SFE}}^{f}$ to do so, then it returns $f(x_1, \ldots, x_n)$ to all players.

Figure 3.1: Ideal functionality for circuit evaluation

The reason why we formalized a delivery in advance step is that, as we will see in 5.1, the adversary can make the functionality abort after this step when more than $n/2$ players are adversaries. We will then discuss solutions to prevent this.

3.1.2 Efficiency issues

As described in Figure 3.3, the protocol against semi-honest adversaries requires each field element input to be shared as n field elements. Then each hard multiplication requires n^2 field elements exchanged, because of the resharing step.

On the other hand, making this protocol resilient against active adversaries as explained in §3.3.2 is much costlier. Each share of a secret must itself be committed, which by Figure 3.7 requires n^2 shares per commitment. In addition, the $t < n/3$ adversarial players can force their shares to be *broadcasted* in Figure 3.7. Using the broadcast protocols of [CW92, BGP92] with complexity n^2 and at worse $t < n/3$ rounds, this further multiplies the total message size by $O(n^2)$ and adds $O(n)$ rounds[18]. At last, one multiplication requires two resharing processes[19], which multiplies by a factor linear n the total message size. Finally we have a message size of $O(n^6)$ for one multiplication, which is comparable to the state of the art obtained in 1992 by [Bea92] with different techniques.

Fortunately, recent progress on the *message-size* for processing shared secrets has firstly been made by [HMP00] in 2000 with n^3 message-size per multiplication. Then [DN07, BTH08] achieved in 2008 a linear size in n per multiplication when the circuit is horizontal (all gates in parallel), but still in n^2 when the circuit is vertical (maximum depth). See also [CDN, §8.7]. *An orthogonal improvement direction consists in packing more than one secret in n shares* (e.g. by considering also the evaluation of polynomials at 1 and not only 0). This naive approach

[18] A publicly cheating committer could further slow down the protocol, but we do not take this into account

[19] [CDN, p110] for multiplication of committed values, plus the resharing step of Figure 3.3

needs to assume one less malicious adversary per additional secret packed. But fortunately [DIK10] could (nearly completely) remove this assumption, at the cost of a computational overhead independant of n. It remains that such packed shared secrets can be only processed by the same gates in parallel (SIMD circuits): see [GIP15] for a discussion.

Nevertheless, it remains an issue with the *number of rounds* for processing shared secrets. Indeed, we stressed that $t < n/3$ hidden adversaries can cause disputes in Figure 3.7 and slow the process down to two more rounds. Each of these rounds involves broadcasts, which themselves decompose in $t < n/3$ rounds, so the round complexity explodes. This possibility of slowdown also occurs in the classical Byzantine consensus [LSP82]. Detecting and deterring such malicious behavior thus seems desirable: see §5.4 and §5.5 for solutions.

The issues of malicious behavior and the bound of n message size per multiplication seem linked. Indeed as stressed by [CDN, §8.6], when all but one players behave maliciously, it is difficult to imagine a solution where this player would have to work less than computing the whole circuit himself.

3.2 Perfect security with $t < n/2$ semi-honest adversaries

The main tool needed is *secret sharing*. Consider a secret a, f_a a polynomial of degree less than t such that $f_a(0) = a$, and suppose, to make it simple, that $1, \ldots, n$ are distinct values in the finite field $\mathbf{F}$. Then note

$\langle a\,;\, f_a \rangle_t$ the distributed state where P_1 holds $f_a(1)$, P_2 holds $f_a(2)$, ..., P_n holds $f_a(n)$.

Then $t+1$ players who put in common their shares $f_a(i)$ can interpolate the polynomial f by Lagrange's interpolation theorem, and thus find its value at zero. Let us formalize this tool: for all integer n, there exists a set of coefficients $\mathbf{r} = (r_1, \ldots, r_n)$ in the field $\mathbf{F}$, such that for any polynomial h of degree $\leq n-1$ it holds that $h(0) = \sum_{i=1}^{n} r_i h(i)$. Finally let us stress that in the previous situation where f_a is of degree $\leq t$, then the knowledge of t shares doesn't learn anything about the secret value $a = f_a(0)$: it is *uniformly distributed*. So that t colluding players don't learn anything about the secret: this is Shamir's secret sharing scheme.

The simulator for $\pi_{\text{mult-passive}}$ is described later in Fig. 3.6 (see also [CDN, §3.3 p43]) and is very easy to describe: it just needs to send t random shares to the adversaries on behalf of each honest player P_i, pretending that these are shares of f_i. Indeed, as t shares of a polynomial of degree t give no information on its value at zero, adversaries are unable to distinguish this from a real-world execution. On the other hand, the simulator for the opening phase, also described later in Fig. 3.6 is a bit more involved: from the value of the final result y and t shares y_i of it, the simulator interpolates the shares of honest players then send them to the adversaries. So that they receive values consistent with the output y and their shares.

3.3 Perfect security with $t < n/3$ adversaries

3.3.1 Preliminary: need to implement a broadcast channel for $t < n/3$.

We will need to use a functionnality called *broadcast*, described in figure 3.4 and similar to a loudspeaker: when a player P_i says something in it, then all players P_j *have the guarantee to hear the same message*. Indeed without broadcast, players do not have the guarantee that a malicious player sends the same message to every players, when he claims to do so.

It is possible to unconditionally implement broadcast (so assuming only secure point to point channels and synchronous communications) for $t \leq n/3$ adversaries: [LSP82] or [CDN,

π_{SFE}

Input sharing Each player P_i holding input x_i chooses a random polynomial f_{x_i} conditioned to $f_{x_i}(0) = x_i$, and distributes the shares $f_{x_i}(1), \ldots, f_{x_i}(n)$ to players $P_1, \ldots, P_n$, resulting in the state $\langle x_i \,;\, f_{x_i} \rangle_t$

Computation the shared secrets will go through addition and multiplication gates, such that an invariant is preserved: every output of these gates is itself a shared secret $\langle b; f_b \rangle_t$ with a polynomial f_b of degree t:

1) addition: the players hold $\langle a \,;\, f_a \rangle_t$ and $\langle b \,;\, f_b \rangle_t$, they add their shares to obtain $\langle a + b \,;\, f_a + f_b \rangle_t$
2) multiplication by a public constant λ: the players hold $\langle a \,;\, f_a \rangle_t$, they multiply their shares by λ to obtain $\langle \lambda a \,;\, \lambda f_a \rangle_t$.
3) multiplication between two variables *(hard)*: naively multiplying the shares would result in evaluations of a polynomial $f_a f_b$ of degree $2t$, which violates the invariant. Thus see figure 3.3 for how to circumvent this.

Output At this point, the players hold a shared secret $\langle y \,;\, f_y \rangle_t$, which is the result of the computation, and want to learn its value. Each player P_j sends $f_y(j)$ to each P_i, who uses Lagrange interpolation to compute $y = f_y(0)$ from $f_y(1), \ldots, f_y(t+1)$ or any other $t+1$ points.

Figure 3.2: Circuit evaluation algorithm for $t < \frac{n}{2}$ honest but curious adversaries

$\pi_{\text{mult-passive}}$

Input: Two shared secrets $\langle a \,;\, f_a \rangle_t$ and $\langle b \,;\, f_b \rangle_t$, with degree t
Output: A secret sharing of the product $\langle ab \,;\, h \rangle_t$, with degree t

1) The players compute $\langle a \,;\, f_a \rangle_t \cdot \langle b \,;\, f_b \rangle_t = \langle ab \,;\, f_a f_b \rangle_{2t}$
2) Define $h = f_a f_b$. Then $h(0) = f_a(0) f_b(0) = ab$, and the parties hold $\langle ab \,;\, h \rangle_{2t}$; that is, P_i holds h(i). Each Pi chooses a random polynomial f_i of degree t that takes value $h(i)$ at 0, and distributes that way $\langle h(i) \,;\, f_i \rangle_t$.
3) Note that $deg(h) = 2t \leq n - 1$. Let $\mathbf{r}$ be the recombination vector, that is, the vector $\mathbf{r} = (r_1, \ldots, r_n)$ such that $h(0) = \sum_{i=1}^{n} r_i h(i)$ for any polynomial h of degree $\leq n - 1$. The players compute

$$\sum_{i=1}^{n} r_i \langle h(i) \,;\, f_i \rangle_t = \sum_{i=1}^{n} \langle r_i h(i) \,;\, r_i f_i \rangle_t$$
$$= \langle h(0), \sum_{i=1}^{n} r_i f_i \rangle_t = \langle ab \sum_{i=1}^{n} r_i f_i \rangle_t$$

Figure 3.3: Multiplication algorithm for $t < \frac{n}{2}$ honest but curious adversaries

$\mathcal{F}_{\text{broad}}$

Input: Player P_1 sends a message m to $\mathcal{F}_{\text{broad}}$.
Output: All players receive message m from $\mathcal{F}_{\text{broad}}$.

Figure 3.4: Broadcast functionality

th 5.9]. It requires a number of rounds equal to the number of adversaries corrupted: see [CDN, p90] .

3.3.2 Principle: upgrading the protocol with perfectly secure commitments

In a nutshell, players will "outsource" all operations performed in algorithm $\pi_{\text{mult-passive}}$ (including evaluation and resharing), to an ideal enhanched committing functionality $\mathcal{F}_{\text{Com}}$.

The point is that $\mathcal{F}_{\text{Com}}$ performs an operation only il all players publicly order her to do so. Thus, as long as there is at least one honest player, she can but only follow the instructions of $\pi_{\text{mult-passive}}$, or abort the process because of a publicly dishonest player. In particular in the resharing phase, dishonest players will commit to coefficients of the resharing polynomial f_i they choose. Evaluation of this resharing polynomial at $1, \ldots, n$ is then a linear computation performed by the commitment functionality —and likewise for the opening of $f_i(j)$ to every other player P_j.

This approach is from Cramer, Damgaard and Maurer [CDM00]. As stressed in [CDN, §5.6], the original approach of [MBOW88] is faster but offers less modularity (as using more modern secret sharing, commitment or signature schemes).

Let us recall that a *commitment is like a safebox in a public place, whose owner keeps the key.* Assume that Bob wants to convince Alice that he has made a choice and will stick to it. But Bob doesn't want to reveal that choice right now. So Bob writes his choice on an enveloppe and deposits it in a closed safebox, that he puts in Alices' home. So Alice can't open it, and Bob can't change the letter inside. One year later, Bob comes back and open the safebox in front Alice: thus she learns Bob's choice.

As we will want here that commited values be added and even multiplied (fully homomorphic commitment), we describe in Figure 3.5 an ideal commitment agent with these additional features (straight from [CDN, §5.2]).

From now on, players will commit to their shares of secret. We note such an enhanced shared-secret:

$$\langle\!\langle a, f \rangle\!\rangle_t \text{ ,which means} \{[\![f(1)]\!]_1, \ldots, [\![f(n)]\!]_n\}$$

They will execute the passive algorithm in figure 3.3 by publicly outsourcing every operation to $\mathcal{F}_{\text{Com}}$ (addition, multiplication, resharing).

The consequence is that $\mathcal{F}_{\text{Com}}$ will necessarily follow the rules of algorithm in figure 3.3. Indeed to deviate from this would require (every) honest players to tell $\mathcal{F}_{\text{Com}}$ to do something wrong (by construction of $\mathcal{F}_{\text{Com}}$). Which is impossible, by definition of honest players. A adversarial player P_i can still make an operation of $\mathcal{F}_{\text{Com}}$ abort but then, as described in figure 3.5, all players will be aware that he made this happen.

In conclusion, as described in [CDN, §5.3]:

- either all players follow the rules, as if adversaries were passive;

$\mathcal{F}_{\text{Com}}$

Commit. P_i (the Committer) holds a secret value a. When all honest players $(P_j)_j$ ask $\mathcal{F}_{\text{Com}}$ to do so, then the secret a is committed. It is stored in the form $(a, cid(a), i)$ where $cid(a)$ is a public *commitment identifier* and i refers to the (publicly known) player P_i who commited it. We note this internal state $[\![a]\!]_i$, sometimes forgetting the i when clear.

P_i *can also make the commitment* abort if he is an active adversary. Then all players learn that he made this happen.

Public commit. Here a is known by all the players before being commited, so the commitment can be seen as merely putting a on a tamperproof bulletin board.

Open. When all honest players $(P_j)_j$ ask $\mathcal{F}_{\text{Com}}$ to reveal the value stored under the identifier $cid(a)$ (a is still unknown to them), then $\mathcal{F}_{\text{Com}}$ does so.

The committer P_i *can also make the opening* abort if he is an active adversary[a] . Then all players learn that he made this happen.

Designated open. Same, but a is only opened to a particular player P_j

Add. When all honest players ask $\mathcal{F}_{\text{Com}}$ to add two values $[\![a]\!]_i$ and $[\![b]\!]_i$ committed by the *same* player P_i (players only know the public identifiers $(cid(a), i)$ and $(cid(b), i)$), then $\mathcal{F}_{\text{Com}}$ creates the internal state $[\![a+b]\!]_i$.

Mult. by constant. Similarly, when honest players ask $\mathcal{F}_{\text{Com}}$ to multiply $[\![a]\!]_i$ by a *publicly known* constant λ, then $\mathcal{F}_{\text{Com}}$ creates the internal state $[\![\lambda a]\!]_i$.

Some advanced (and costlier) functionnalities for $\mathcal{F}_{\text{Com}}$, that can be implemented from the previous ones:

Transfer. When all honest players ask $\mathcal{F}_{\text{Com}}$ to transfer $[\![a]\!]_i$ from P_i to P_j, then he creates the internal state $[\![a]\!]_j$ [b].

The committer P_i *can also make transfer* abort if he is an active adversary. Also either sender P_i or receiver P_j can make the value of $[\![a]\!]_i$ opened to everyone if one of them is an active adversary[c](allowed "leakeage" of information by an adversary).

P_j *can also make the transfer abort* if he is an active adversary. Then all players learn that he made this happen.

Mult. When all honest players ask $\mathcal{F}_{\text{Com}}$ to multiply two values $[\![a]\!]_i$ and $[\![b]\!]_i$ committed by the *same* committer P_i, it creates the internal state $[\![a+b]\!]_i$.

The committer P_i *can also make the multiplication* abort if he is an active adversary. Then all players learn that he made this happen.

[a] By contrast, *verifiable secret sharing* doesn't enable a dishonest committer P_i to abort the opening. This additional security property actually holds for the protocol that we are going to see because it is based on Shamir secret sharing scheme, as in Rabin and Ben Or [RBO89]. This is no longer true for arithmetic secret sharing: see end of [CDN, §5.6]

[b] In particular, P_j will then be able the make opening of $[\![a]\!]_j$ abort if he is an active adversary.

[c] Notice that the definition in [CDN, 5.2] doesn't allow a corrupt P_i to leak a. But we do it here, since the two protocols proposed in [CDN, 5.2.1] enable this (here P_i can force alone to go in the "exception handler").

Figure 3.5: Ideal commitment agent

- or a player P_i makes an operation abort .He is then excluded from the protocol by the honest majority, and the protocol goes on without him.

Let us describe formally the simulator sketched at the end of §3.2, that also suits to the active adversary setting (see [CDN, §5.3 p116]):

3.3.3 Realizing a perfectly secure commitment protocol, and why the limit $n/3$

The idea is to have the commiter P_1 secret-share the value a he wants to commits to between the players, but with extra redundant information ensuring unicity of a. So instead of using a polynomial $f_a(X)$ in degree t and distributing its values at $1, 2, \dots n$, he will choose a *bivariate symmetric polynomial*

$$f_a(X,Y) = \sum_{u=0}^{t}\sum_{v=0}^{t} a_{u,v}X^uY^v$$

evaluating to a at zero; and distribute its evaluations in $Y = 1, 2, \dots n$, which are *univariate polynomials*:

$$f_k(X) = f_a(X,k) = \sum_{u=0}^{t}\Big(\sum_{v=0}^{t} a_{u,v}k^v\Big)X^u$$

In particular the values $f_k(0)$ are a classical secret sharing of a with polynomial $f_a(0,Y)$ of degree t.

The protocol consists in having players check that the information shared is consistent. In particular by symmetry of f_a, we should have for any two players P_k and P_j that

$$f_k(j) = f_a(k,j) = f_a(j,k) = f_j(k)$$

Let us formalize the commitment protocol in figure 3.7 (from [CDN, p 124]).

Let us finally explain the reason for the bound $t < n/3$ as in [CDN, p 121]. Let $f(X) = f_a(X,0)$ be a univariate polynomial of degree less than t, as the one obtained in step 8) of figure 3.7, such that $f(0) = a$ is the secret. Consider the vector of shares:

$$\mathbf{s}_f = \big(f(1), \dots, f(n)\big)\ .$$

Let now $\mathbf{e}$ be an arbitrary vector of hamming weight $\leq t$, that represents the errors introduced by the $t < n/3$ adversarial players. Define accordingly

$$\widetilde{\mathbf{s}_f} := \mathbf{s}_f + \mathbf{e}$$

the vector of shares with errors. Then we claim that a is still uniquely determined by $\mathbf{s}_f$. In fact the whole polynomial $f(X)$ is determined by $\mathbf{s}_f$. Indeed suppose that there exists another polynomial $g(X)$ of degree at most t that also explains the vector $\mathbf{s}_f$ with at most t errors. Let $\mathbf{u}$ the vector of errors that accounts for this:

$$\begin{aligned}\widetilde{\mathbf{s}_f} &= \mathbf{s}_g + \mathbf{u} \text{ subtracting both equalities yields:}\\ \mathbf{s}_{f-g} &= \mathbf{s}_f - \mathbf{s}_g = \mathbf{u} - \mathbf{e}\end{aligned}$$

But by assumption, the vector $\mathbf{u} - \mathbf{e}$ is of Hamming weight smaller than $2t$. So there exists more than $n - 2t \geq t + 1$ coordinates at which $\mathbf{s}_{f-g}$ is zero. This means that the polynomial $f - g$ of degree $\leq t$ vanishes at more than $t + 1$ values, and thus is zero.

$\mathcal{S}_{\mathrm{SFE}}$

Initialization The simulator $\mathcal{S}$ creates virtual honest players $P_{t+1}, \ldots, P_n$, and arbitrarily chooses their inputs (for example $\widetilde{x_i} = 0$ for all of them).

Input sharing The simulator interpolates[a] the inputs x_i of adversarial players $P_t, \ldots, P_n$ from the shares they sent to him (and addressed to the virtual honest players). These shares are consistent since adversaries were forced to behave honestly by using $\mathcal{F}_{\mathrm{Com}}$). $\mathcal{S}$ then passes these inputs x_i to the functionality $\mathcal{F}^f_{\mathrm{SFE}}$. The real honest players (that $\mathcal{S}$ doesn't interact with) also pass their inputs to $\mathcal{F}^f_{\mathrm{SFE}}$.

Computation During the computation, the simulator $\mathcal{S}$ acts as if virtual honest players $P_{t+1}, \ldots, P_n$ followed the protocol. In fact the only moment when $\mathcal{S}$ interacts with adversaries is during the resharing round of the multiplication protocol. There, as stressed by [CDN, Observation 1 p40], the simulator also could well send random shares to the adversaries during this step.

Opening At this stage, since the adversaries were forced to follow protocol π_{SFE} because everything was done by $\mathcal{F}_{\mathrm{Com}}$, we are in a state $\langle \widetilde{y}\,; f_{\widetilde{y}} \rangle_t$ where all the players, both adversaries and virtual players, hold consistent shares $\widetilde{y_i} = f_{\widetilde{y}}(i)$ of the output $\widetilde{y}$ of the circuit. The problem is that this output is not the one y of the real execution, since it $\widetilde{y}$ was obtained from dummy inputs $\widetilde{x_i}$ of the virtual honest players. So the simulator will compensate for this. In detail:

1) $\mathcal{S}$ asks $\mathcal{F}^f_{\mathrm{SFE}}$ to evaluate the function (remember that $\mathcal{F}^f_{\mathrm{SFE}}$ was given all the inputs at input sharing step). When the real honest players also ask her to do so, then $\mathcal{F}^f_{\mathrm{SFE}}$ returns the output y to $\mathcal{S}$ (and to the real honest players).
2) $\mathcal{S}$ has thus $t+1$ shares $y, \widetilde{y_1}, \ldots, \widetilde{y_t}$, from which it can interpolate the unique polynomial f_y of degree t evaluating to y at zero and to $\widetilde{y_i}$ at i.
3) $\mathcal{S}$ then derives $n-t$ consistent shares $y_j = f_y(j)$, $j = t+1, \ldots, n$ that he sends to the adversaries on behalf of the honest players. The adversarial players are then able to interpolate y from these shares.

[a] Notice that in the equivalent presentation of [Can01], the simulator impersonates the adversaries to the environment $\mathcal{E}$. But this doesn't imply that $\mathcal{S}$ will learn the adversaries' inputs directly from the environment $\mathcal{E}$. So $\mathcal{S}$ also needs to interpolate them in this setting.

Figure 3.6: Simulator for multiplication and opening, which holds against $t < n/2$ semi-honest adversaries (§3.2) and against $t < n/3$ active adversaries, proving Theorem 3.1

π_{Commit}

1) When all players ask him to do so, the committer P_i samples a bivariate symmetric polynomial $f_a(X, Y)$ such that $f_a(0, 0) = a$. She sends the polynomial $f_k(X) = f_a(X, k)$ to each P_k.
2) Each P_j computes $\beta_{k,j} = f_j(k)$ and sends $\beta_{k,j}$ to each P_k.
3) Each P_k checks that $\deg(f_k) \leq t$ and that $\beta_{k,j} = f_k(j)$, for $j = 1, \ldots, n$. If so, he broadcasts success and, if not, broadcasts that there is a dispute with P_i or P_j for each inconsistency.
4) For each dispute reported in the preceding step, P_i samples the correct value of $\beta_{k,j}$.
5) If any P_k finds a disagreement between what P_i has broadcast and what he received privately from her, he knows that P_i is corrupt and broadcasts an accusation against her.
6) For any accusation against her from P_k in the preceding step, P_i broadcasts $f_k(X)$
7) If any player P_k finds a disgreement between what P_i has now broadcast and what he received privately from her, he knows that P_i is corrupt and publicly accuses her.
8) If all the information broadcast by P_i is not consistent or if more than t players have *accused* her, players abort the protocol. Otherwise, players who accused P_i and had a new polynomial $f_k(X)$ broadcast will accept it as their polynomial. Now each player P_k output success and stores $f_k(0) = f_a(0, k) = f_a(k, 0)$ (along with the commitment identifier). In addition, P_i stores the polynomial $f_a(X, 0)$: notice that the values $f_a(k, 0)$ are shares of this univariate polynomial.

Figure 3.7: Commitment protocol

4 *Statistical security* when less than *half* of participants cheat, *conditionned to broadcast or correlated randomness*

4.1 Summary of the section

4.1.1 Theory

Unconditional secure MPC is impossible in general for more than one third participants, because a functionality as simple as broadcast cannot be implemented ([CDN, th 5.10]), even with statistical security.

Now, if one *assumes* access to a broadcast functionality as part of the model, then one can reach the bound $t < n/2$. This was first proved by Rabin and Ben-Or[20] [RBO89] and will be detailed in the §4.2.

Theorem 4.1 ([CDN, Th 5.13])**.** Assuming availability of a broadcast channel*, then any MPC functionality can be implemented with statistical security against* $t < n/2$ *active adversaries.*

We will briefly mention in §4.3 why the assumption on broadcasting can be replaced by assuming access to a functionality called *correlated randomness.*

4.1.2 Efficiency issues

The *size of messages* sent for one multiplication with information-theoretic security grows at least linearly in n and in the *depth* of the circuit, according to [DNPR16] or [Pol16, Corollary 5.7]. See also the latter for a discussion on communication complexity in various models. This lower bound is nearly matched by [BSFO12].

Better than the commitment protocol described below figure 3.7, [GGOR13] provides a verifiable secret sharing scheme (see footnote in figure 3.5) for $t < n/2$ that makes only a *constant number of calls to broadcast* (at most three).

Considering the impact of circuit's depth, which is the bottleneck of FHE: the resilient circuits proposed by [GIP+14, Table 1, Th 1.3 and 1.4] or [GIP15][21] , enable protocols that are not affected by depth, but only by the circuit's overall size $|C|$. This is not surprising since [DNPR16] states that gains from multiplications in parallel can be at most linear in n.

Finally, if using garbled circuits (a priori non UC secure) and a communication scheme centered around a server, then it is possible to achieve a *number of rounds independent of the size of the circuit*: see [DI05][22] , which makes black box use of a pseudorandom generator[23]. and of a consant number of broadcasts. With the same hypotheses, [DI06] furthermore achieves *scalability, i.e. a number of rounds independant from the number of players.* Here it is discussed how to implement correlated randomness from linear codes, a pseudo random generator and broadcast. It is actually claimed that the broadcast assumption could be remove without harming the average performance of the protocol.

Finally, if one doesn't have access to broadcast nor correlated randomness, then [FGMvR02] still proves that for $t < n/2$ adversaries, one can achieve a weaker version of MPC where *output delivery is not guaranteed.* More precisely, although the adversary can neither violate

[20] [FM00] prove that a three-players broadcast is actually enough.

[21] Which has the advantage to be compatible with secret sharing-based protocols

[22] [IKP+16] further compiles this protocol to make the number of rounds independant of the security parameter.

[23] I.e. a functionality that takes as input a seed, and outputs a number following uniform distribution. The same seed will output the same number

correctness, nor privacy, nor learn the result and quit without letting the other parties also learn it[24]; *he can anonymously make the functionality* abort before it delivers its result. [25]

4.2 How assuming access to a broadcast channel enables the bound $t < n/2$

4.2.1 Main idea: signing each message during the commitment protocol

The idea is that in figure 3.7 during step 1), every message $f_k(X)$ of the commitment protocol is now signed by the committer P_i. Thus any player P_k accusing P_i of inconsistency at step 4) from P_j, will be able to prove to all the players that inconsistent messages were indeed sent to him by P_j. As illustrated in [CDN, §5.4 p134], this makes a shorter commitment protocol that now achieves the bound $t < n/2$.

4.2.2 Implementing IC signatures

In figure 4.1 we describe the exact functionality needed: the *information checking (IC) signature*[26] $\mathcal{F}_{\text{IC-sign}}$. We stress that is is not a public key signature, nor does it rely on public key cryptography. Instead they are additively homomorphic signatures, whose binding property only relies on the existence of a majority of honest players.

$\mathcal{F}_{\text{ICSign}}$

Signature A player P_1 (the sender) gives a message m to $\mathcal{F}_{\text{ICSign}}$. When all honest players ask $\mathcal{F}_{\text{ICSign}}$ to do so, then: $\mathcal{F}_{\text{ICSign}}$ outputs "signed" along with an identification number for message m; and also sends message m privately to a particular player P_2 (the intermediary).

P_1 *and* P_2 *can also together make the signing* abort if they are active adversaries. All players will then be aware that they both made this happen.

(Designated) Reveal When all honest players ask $\mathcal{F}_{\text{ICSign}}$ to reveal message m (identified by its identification number), then $\mathcal{F}_{\text{ICSign}}$ outputs m to all the players (or to one single player).

The intermediary P_2 *can also make the reveal* abort if he is an active adversary.

Add and mult by constant Same as in the committment protocol[a]. This can't be aborted by any player.

[a] Such signatures are therefore called *additively homomorphic.*

Figure 4.1: Information checking (IC) signature

[24] This means that the protocol is fair: see [CL14, §1.2]. In this model, an adversary can't aborts just after he has learnt an output and before the other honest players could learn it. But this may occur for an intermediary output, before the whole computation terminates. Plus, the cheater may remain undetected. So this doesn't prevent anonymous early abort, which constitutes a denial of service attack (see below). Thus we won't review this line of research.

[25] See also [CHOR16] who investigate functionalities that are computable without broadcast for $t < n/2$, and even $t < n$ assuming one-way functions

[26] See [IOZG14, Appendix A] for a survey of implementations. Notice also a more efficient signature protocol in [CBO14], involving only one broadcast (by the signing player), but where the signing player is assumed honest.

As in many protocols with statistical security, the implementation described in [CDN, §5.4.1] uses the following flavor of one-time pad:

Definition 4.1 (Message authentification code (MAC))**.** *Let* $\mathbf{F}$ *be the finite field in which messages live. A pair* $K = (\alpha, \beta)$ *of elements of* $\mathbf{F}$ *is called a MAC key.*

Let m *be an element of* $\mathbf{F}$ *(a message), then* $\alpha m + \beta$ *is called the MAC of* m *with key* K*.*

MACs are additively homomorphic for two messages having the same α key.

π_{ICSign}

Signature The sender P_1 gives message m to the intermediary P_2, along with several MACs of it. Separately, P_1, gives one of each MAC keys $K_j)(\alpha_j, \beta_j)$ to every other player P_j.

Check of consistency Players check with challenges that MACs are well formed. Now, everybody has the guarantee that message m is correclty signed, but only P_2 has access to the value of m.

Addition (analogous for multiplication by a constant) P_1 computes the MACs of the sum, P_2 computes the sum of the messages and the sum of their MACs, and finally every other player P_j adds the linear parts of the two MAC keys that he holds (the β_j's).

Opening and proof Later, P_2 discloses m to everybody along with the MACs, such that all players are convinced that the message received was indeed m.

Figure 4.2: Information checking signature protocol

4.3 Assuming access to a source of correlated randomness also enables the bound $t < n/2$

To start with, let us discuss a simpler functionality: *multiparty coin tossing*. This functionality consists in outputing a uniformly distributed value[27] . So from the point of view of the players it is a public random source: see [CD17, KRDO17, SJK+17] for improvements and applications to blockchain consensus for $t < n/2$.

According to the introduction of [BOO10], computing a random value between players would be well known to be in the scope of MPC, and in particular would be perfectly implementable for the bound $t < n/3$ with the techniques described above in §3.3, although no reference explaining this fact is known to us. On the other hand the authors also recalls the naive solution for two players, from the seminal work of [Blu83], and stresses why it produces a bias. Basically this solution consists in having Alice and Bob commit to random values, then open the commitment and make the sum. The problem is that if Alice reveals her commitment first to Bob, then he has the possibility to make the opening fail if he is not satisfied with the value. Upon failure, Alice will then output her sole random bit: summing all cases, Bob has probability 3/4 to win.[28]

[27] As stressed in [FGMvR02, §3], it can be proven that this sole functionality does not help to improve the classical bound of $t < n/3$ for broadcast (and hence for MPC in general).

[28] Notice that in this example, Alice will then be aware that Bob is a cheater: we have achieved *covert*

Hence, [Cle86] has shown that with $t \geq n/2$, then, in any r-round coin-tossing protocol, the malicious parties can cause a bias.

The modern solutions for $t \leq n/2$ quoted above involve *verifiable secret sharing*, that prevent a commiter to make the protocol abort.

Now, the more demanding *correlated randomness functionality*, described in figure 4.3, has a status comparable to broadcast:

- on the one hand, it is impossible to realize it unconditionaly for more that $t < n/3$ adversaries,
- but on the other hand, assuming it given for free (as an assumption of the model) enables the bound $t < n/2$ adversaries.

The last point was proven by [FGMvR02][29] . We describe in 4.3 the general version of correlated randomness, as found informally in the update [FWW04, §3.2], and from which they implement signatures (and thus MPC for $t < n/2$ as in §4.2) from it.

$\mathcal{F}_{\text{CorrRand}}$

Let $(X_1, \ldots, X_n)$ a (fixed) random vector with probability law $P_{(X_1,\ldots,X_n)}$.
Input: Each of the n players, $P_1, \ldots P_n$, asks $\mathcal{F}_{\text{CorrRand}}$ to deliver the output.
Output: $\mathcal{F}_{\text{CorrRand}}$ randomly samples $(x_1, \ldots, x_n)$ following $P_{(X_1,\ldots,X_n)}$ and sends x_i to each P_i.

Figure 4.3: Correlated randomness functionality

[IKM+13] further discusses the topic. E.g. how correlated randomness implements the primitive *oblivious transfer*, described in Figure 4.4, which is itself *complete* in the sense that it enables MPC for $t < n/2$ with statistical security. See [CGS16] for a discussion on to what extent it is possible to implement this functionality in a quantum state. Finally [WW10] gives linear lower bounds on the storage complexity of secure computation with correlated randomness.

$\mathcal{F}_{\text{OT}}$

Input: The *sender* P_1 inputs two values x_1 and x_2 in $\mathbf{F}$. The *receiver* P_2 inputs a bit b.
Output: P_2 receives message x_b.

Figure 4.4: Oblivious transfer

The previous primitive can actually be amortized over several executions with a fixed b (Correlated oblivious transfer). Let us now describe the different flavour of OT, $\mathcal{F}_{\text{COPe}}$, used in preprocessing-based protocols [FKOS15, KOS16].

security as described in 5.4, and actually the paper provides a proof of cheating auditable by an external party. Notice that for $t < n/3$ (or $t < n/2$ with a broadcast channel), then the commitment described in 3.3.2 also identifies the player who makes the opening abort (since only the committer can cause this), so also achieves covert security.

[29] From a three-player correlated randomness instance, they built a broadcast, and thus any MPC for $t < n/2$ adversaries by Theorem 4.1.

$\mathcal{F}_{\text{COPe}}$[a]

Input: The receiver P_2 inputs a fixed $\Delta \in \mathbf{F}$. The sender inputs $x \in \Delta$. (The cost for several executions with different x and a fixed Δ being amortized).
Output: P_1 receives $t \in \mathbf{F}$ and P_2 receives $q \in \mathbf{F}$ such that $q + t = x \cdot \Delta$.

[a]We describe only the semi-honest functionality, which is simpler than the actual functionality $\mathcal{F}_{\text{OT}}$ in the active adversary setting, where the adversary has additional powers. There, if P_B is corrupt he can choose t. If P_A is corrupt he can replace the finite field product $x \cdot \Delta$ by a more complicated transformation applied on Δ, that does not necessarily corresponds to a finite field product.

Figure 4.5: (Correlated-) oblivious transfer with errors

Correlated randomness can today be built from: cryptographic tools as OT or zero-knowledge proofs[30] , or AES computations as promoted in [AFL+16, FLNW17], or noisy communication channels between the players / weakly correlated pieces of information as stressed in [FWW04]. But it could also be obtained from an external dealer. Beaver [Bea97] indeed sets a model with several independant dealers, such that a minority of them colluding with adversary players can't harm the overall randomness. The thesis work [Dow16] also studies (among other issues) what can be done in this "dealer" model.

5 Breaking the honest majority limit with more crypto, and the problem of early aborters

5.1 Summary of the section

Security with abort: As we saw in the last section, when there is no honest majority, unconditional security cannot be achieved. One can still obtain an even weaker version of MPC than mentionned at the end of §4.1: security with abort. This means that the adversary can make the protocol abort in 3.1 after it has learned the output, before it is output to honest players[31]

§5.3 *Identifiable abort*: A solution to fix this would be that honest players agree on at least on the identity of an adversary player when early abort happens. This intermediary security property is defined in [IOZG14]. *This property is hard to reach*, since [IOS12] claims (without proof) that identifiable abort is impossible to obtain for $t < n$ adversaries, even assuming trusted access to every pairwise functionalities and a broadcast channel. But solutions exist provided *correlated randomness*, as proposed in [IOS12, IOZG14].[32].

§5.2 *Preprocessing-based protocols*: such protocols decompose themselves in two phases. Firstly, players communicate without needing to know their inputs, in a so-called *offline phase*. Technically, they create correlated randomness between them as described in figure 4.3. When this phase is finished, and when players learn their inputs, then the second phase can start: this is thus called *online phase*. It is typically much faster than the offline phase. The practical use cases are thus when players need to perform fast computations and know

[30]See §5.2: the protocols [KOS16, KPR18] preprocess a flavour of correlated randomness (Beaver triples).

[31]For example, assuming access to oblivious transfer provides this relaxed security: see [IPS08] or [KOS16] for a recent preprocessing-based implementation.

[32]With cryptographic asumptions and removing the UC security, then [GMW, Gol06] also provides this functionality but is not UC secure: see [DO10]

in advance the other players, so that they can perform the offline phase before. Another advantage is that the identifiable abort security property can be efficiently obtained during the online phase.

§5.4 *Covert security*: Instead of trying to achieve both security against malicious adversaries and identifiable abort, which may slow down protocols a lot, a recent orthogonal direction consists in designing algorithms that identify misbehaving players (or just early aborters) with a non negligible probability.

§5.5 *Public auditability*: is a security feature that issues a proof that the protocol was correctly executed or not, which can be checked by any external person. In costlier protocols, this proof also identifies players who misbehaved during the protocol.

Before we detail these additional security properties in the next subsections, we would like to point that most of the protocols studied in the litterature do not satisfy them: they satisfy only security with abort. Nevertheless, being aware of them seems still important. Indeed, since they are more investigated, they have a priori optimized performances. So that a possible direction for research would be to *transform* them as explained in [IKP+16], in order obtain the desired security properties —in particular identifiable abort. A comparison of security with abort protocols can be found in [GIP+14, Table 1][33] (whose results are nailed down by [GIW16]) or [GIP15][34] or the generic active-to-passive compiler of [DO10]. Let us finally mention a study [NR17] about the —less known— overall *computational* cost of security with abort protocols.

5.2 Preprocessing for a fast and secure online phase

5.2.1 Principle (the semi-honest case)

We saw that *linear operations* on linearly shared-secrets, namely addition and scalar multiplication, are easy, because each player simply performs it on his share. On the contrary it is hard to multiply two shared secrets. The trick of Figure 5.1 is often credited to [Bea92]. It enables to replace each hard multiplication with a couple of (easy) scalar multiplications and additions —but provided a long preprocessing phase (or access to a trusted *correlated randomness* dealer).

The input phase is described in [CDN, p176], where the players share their inputs $\langle x \rangle$ and $\langle y \rangle$ *with the help of extra preprocessed random variables.* Addition and multiplication by a constant are obvious.

5.2.2 Protocols robust against $t < n$ active adversaries, with possibility of abort

We present today's most used online phase, which originates from e.g. [BDOZ11]. It resists against a dishonest majority thanks to *consistency checks with MACs* (see definition 4.1), which work as follows:

An fixed random global MAC key is also additively secret-shared among the players in the offline phase $\langle \alpha \rangle$, $\alpha = \alpha_1 + \ldots \alpha_n$.

Then each share x_i of an additively shared secret $\langle x \rangle$, $x = x_1 + \cdots + x_n$, is packed with a

[33] Notice in particular a solution with correlated randomness provided by a corrupt dealer, in $n^2|C|$ communication complexity.

[34] Which has the advantage to be compatible with secret sharing-based protocols

$\pi_{\text{PreprocessedMult}}$

Preprocessing A functionality distributes to the players three perfectly random shared secrets: $\langle a \rangle$, $\langle b \rangle$ and a sharing of the product: $\langle c \rangle$, $c = ab$.

Multiplication of shared inputs They compute (sharewise) the additions

$$\langle \epsilon \rangle = \langle x - a \rangle \text{ and } \langle \delta \rangle = \langle y - b \rangle$$

Then they open the values of ϵ and δ. Finally they compute (sharewise) the linear multiplications and additions:

$$\langle z \rangle = \epsilon\delta + \epsilon\langle b \rangle + \delta\langle a \rangle + \langle c \rangle \ .$$

Figure 5.1: Multiplication of additively shared secrets, with preprocessing

share α_i of $\langle \alpha \rangle$ and also with a share Δ_i of the MAC of the secret:

$$\text{MAC}(x) := \alpha x = \Delta_1 + \cdots + \Delta_n.$$

Thanks to the additive-homomorphic property of MACs, the global MAC key α is not changed throughout the circuit evaluation if players behave honestly (see below). But when malicious behavior is detected then this happens (see below). So every shared data is also matched with the secret-shared MAC key that applies to it. Every shared data is thus (unfortunately also) noted:

$$[\![x]\!] = \{\langle x \rangle, \langle \alpha \rangle, \langle \alpha x \rangle\} \ , \tag{1}$$

In particular the preprocessing functionality is required to secret-share every random variable $[\![r]\!]$, and every triple $([\![a]\!], [\![b]\!], [\![c]\!])$ in the form (1). Notice that one secret-shared (preprocessed) random variable is required for each time a player shares an input.

Since this form behaves well under linear operations (provided adequate addition and scalar multiplication rules on shares) and since only linear operations should be performed during the online phase, then the MAC of the final result $\langle z \rangle$ should be again equal to $\alpha.z$. To check that this is true, players compute then open

$$\langle \alpha \rangle.z - \langle \alpha.z \rangle \tag{2}$$

which should be zero. Notice that no information on the actual value of α has been disclosed if the result is indeed zero. Also, several MACs done with the same α can be checked at once: just take a random linear combination of them and perform the previous test (2) on it.

Theorem 5.1 ([CDN, 8.7][35])**.** *Taking the preprocessing step for granted, then the previous protocol provides MPC with* statistical *security against $t < n$ adversaries* with possibility of early abort.

Furthermore, no secure point to point authenticated messages are needed anymore.

See subsections 5.4, 5.3 and 6.4 for very recent progress and implementations. See also [LSSV16] for a number-of-rounds-efficient preprocessing-based protocol, using a tradeoff between somewhat homomorphic encryption and garbled circuits.

Replacing *statistical* security by perfect security is not possible (even with more assumptions): see [IKM+13, Th 4].

[35] or [IOS12, §5] or see [PW92] for the general setting

5.3 Identifiable abort

The ability to detect adversaries who make the protocol abort was formalized in [IOZG14][36].

[BOS16] solves the problem with a precomputation-based protocol, where the identifiable abort security property also holds in the offline phase. On the contrary, [SF16, CFY17] settle this security level for the online phase only.

Notice finally that [SF16] allow only every honest player to identify at least one cheater, but without consensus between honest players.

5.4 Security against *covert adversaries*: ones that do not want being seen cheating or aborting

5.4.1 Security notion

Aumann and Lindell [AL10] introduced the following notion, mainly known as UC-*covert security*[37] . It deters adversaries that don't want to risk being caught cheating or making the protocol abort (possibly taking the result just before, or simply to make a denial of service attack):

Definition 5.1 (Explicit Cheat Formulation (ECF))**.** *We say that a protocol has ECF if when a player P_i cheats, then there is a* nonnegligible probability ϵ *(the* deterrence factor*) that all the honest players will identify him as a cheater.*

In particular when there is a majority of honest players, then if a majority of players accuse P_i to cheat, this proves that P_i really is a cheater.

As stressed in [AL10, p290], we ask that an adversary making the protocol abort must be identified by all the players[38]*.*

Such solutions thus seem desirable when players don't want to be seen cheating by a business partner (e.g. satellite collision between states: see e.g. [CFY17]). Or if one wants to create a *reputation system between players*[39] , with possibly financial rewards or penalties.

It can allow faster protocols than fully secure against against adversaries. To illustrate this, [DKL$^+$13, table 1 vs. table 2] builts a preprocessing-based algorihm (now superseeded) that achieves covert security 6 times faster than against active adversaries, the gap being even greater in [DKL$^+$12, table 4][40].

5.4.2 State of the art for covert security

Against $t < n/2$ adversaries: [DGN10] explains how converts a passive secure protocol into a covert secure one, with deterrence probability 1/4. It consists in running twice the passive protocol while committing to every messages and inputs, and then opening the dummy

[36] See an *identifiable secret sharing* in [IOS12] for an arbitrary number of adversaries (and applications to MPC). Also UC commitments with identifiable abort and dishonest majority: in [IOZG14, §4] conditionned to correlated randomness (although not required in [IOS12, 4.2]), and [DKL$^+$13, A.] with just randomness

[37] We add the UC prefix, although seldom used, because the original definition is really an equivalent of UC security, with probability of failure. In particular it admits a covert universal composability theorem

[38] Relaxed by [DGN10, §2]

[39] [ALZ13]): from the reputation marks, they deduce subsets of players that should contain a honest majority except with small probability.

[40] This belief is contested by [KPR18]. But their protocol doesn't achieve identifiable abort, so doesn't deter early aborters, contrary to *covert secure* protocols

execution (cut and choose). This idea is specified by [LP14, LPJ17], which also bring the improvement to work over any ring (not only finite fields). The authors implemented it only on the platform Sharemind [Cyb18] (for three players).

Against $t < n - 1$ adversaries: [NME13] converts any passive protocol to a UC covert secure one. Due to the impossibility to implement an unconditional UC commitment without honest majority, the authors use instead cryptographically-binding functions for this purpose. The difficulty is then the following. Recall that the simulator of Figure 3.6 (or the one of commitment), when doing the opening step 2), could adjust the shares of virtual honest-players such that adversaries receive messages consistent with the output (it performed a linear shift of the polynomial). In this setting it is no longer possible to do adjust messages, in polynomial-time, so as to make them consistent with encrypted data. Therefore the authors use a weaker primitive, called "lossy encryption scheme" (the authors' implementation of which being based on Pallier encryption).

Against $t < n - 1$ adversaries, with preprocessing: [DKL$^+$13] that also sets for covert security even in the preprocessing phase, and is based on lattice-cryptography. Contrary to the previous one, based on a cut-and-choose zero-knowledge proof of correct execution, the improvement [DKL$^+$12] uses Schnorr-like zero knowledge proofs (as generalized [CD09]). The paper reports on implementations and benchmarks[41].

5.5 Public auditability

5.5.1 Public blame of cheaters

In the previous framework of covert security, an honest player will not actually be able to prove anyone else that the person they identified was indeed a cheater[42].

This has limits, especially from the point of view of a small player seeing a big player cheating.

A fix could be to let players decide to vote for cheaters at majority. But still one cannot avoid that a majority of players be dishonest and falsely accuse an honest player.

Thus it could be desirable to exhibit a proof of misbehavior, that would be readable by any external person (a "judge"): [AO12] for the definition.

Solutions with garbled circuits are proposed for two players in [KM15, Mal16]. The main building block is an (efficient) oblivious transfer (see figure 4.4) enhancement, where the message sent to the receiver is augmented with a *signature* that proves that it was sent by the sender during this execution.

As regards preprocessing-based protocols, a solution is sketched by [BOS16, chap 8]. By contrast, [CFY17] settles this only for the online phase (after the players have been asked for their inputs). This could be sufficient in real use-cases.

[41] Let us remind the practical performance parameters used when benchmarking algorithms (from e.g. [SAM$^+$17]): We define latency as the time it takes to perform one operation. For example, we define the latency of an AES call as the time it takes all the parties to compute this single AES call. The throughput of an operation is measured as the number of operations that can be performed in one second. This has to do on how well we can parallelise the operation. For example, the number of AES calls we can make in one second.

[42] Let us mention that the previous protocols would easily be made publicly auditable if honest players were willing to disclose their inputs to a judge: see [BOS16, chap 8].

5.5.2 Public proof of correctness of the result

A close notion is "public verifiability", where the computation comes along with a public proof of correct execution. If the execution suffers from cheating, then the proof will report it, but will not identify which participant cheated. See [CBO14] for a construction based on additive crypto commitments.

This is thus interesting from the point of view of a client asking a cluster of servers to perform a MPC. For example data mining of proprietary databases: see [ZNP15].

Another example is [CD17], where public randomness is produced by a multiparty computation between an external cluster of servers: assuming honest majority[43] among them, then their output is guaranteed by a publicly verifiable certificate.

5.6 When synchronicity or completeness of the network cannot be achieved

5.6.1 Asynchronous MPC

In this model, messages are no longer simultaneously delivered in rounds: each message can suffer arbitrary delay. On the other hand, players do not wait the end of a round to start sending messages for the next round. [CGHZ16] defines asynchronous model with eventual delivery in the UC framework and introduces a notion of asynchronous round complexity. It then provides a constant-round protocol for $t < n/3$ adversary players, provided that garbled circuits have been jointly produced by players in a preprocessing phase. Black-box one-way functions and a pseudorandom functionality are also used, very similarly to what was described in §4.1.2.

Under the additional assumption that there exists an upper-bound on the delivery time of messages, [CGLR17] (re)explains how to implement broadcast unconditionally for $t < n/3$, and applies it to implement a generalistic consensus functionality (motivated blockchains).

5.6.2 Server-based one-pass protocols

[HLP11] introduced the concept of "server-based one-pass protocol". Here, to the parties $P_1, \ldots, P_n$ (the "clients"), the model adds a possibly adversarial party P_{n+1} called the "server". Each client P_i is asked to interact only once with the server, independently of the other clients. At the end, the server knows the output of the computation $f(x_1, \ldots, x_n)$.

Each interaction with the server consists in a two-party computation between P_i and P_{n+1}. The secret input of P_i is x_i. Whereas the secret input of P_{n+1} is the output of its successive 2PC $\left(f_j\right)$ with all the previous clients: call it y_i. The output of the server is:

$$y_{i+1} = f_i(x_i, y_i).$$

The functions f_i are tailored so that the coalition of the server with a clique of players learn as little as possible about the inputs x_i (see [GMRW13, definition 2]). The desirable security notion for the overall protocol being, on its turn, stated in [GMRW13, definition 3]. This paper achieves the two previous objectives for a wide range of possible functions f.

[43] Because of the impossibility result of [Cle86]

5.6.3 Non-interactive MPC

[BGI+14] introduced the notion of non interactive protocols, where players need not send messages except at the end of the protocol. The model requires a preprocessing functionality to send the players P_i a particular instance of correlated randomness r_i at the beginning of the protocol. Then, each player P_i does a local computation from r_i and his input x_i, and outputs a final result m_i. The m_i can be then gathered by a server to reconstitute the output (or every player could do this, but this is not prescribed by the model). This scheme is similar to preprocessing-based protocol, except that players need not even share their inputs at the beginning. The drawback is that [YO16] showed that even in the semi-honest model, achieving security requires that the sum of the size of the outputs m_i grows linearly with the size of the *set* of messages x_i (e.g. the cardinality of the finite field $\mathbf{F}$), so is *exponential* in the bit-length of x_i. [OY16] proposed a protocol achieving this lower bound.

6 Versatility of use-cases and best general protocols

In this Section, we show various uses and instantiations of MPC, which deviate from the basic definition, and show the versatility of applications of MPC.

6.1 MPC and encryption (AES)

The computation of AES encryptions is a standard benchmark for most authors. Let us explained why evaluation of encryption is relevant in practice. Imagine relevant shares of secret data are held by several players of a MPC protocol: depending on the models (see above), the players learn no information on the data. Then a computation can be performed on the shares, and the user wishes to learn the result of the computation on plaintext data. One solution would be that the user collect shares himself, and rebuild the result.

This solution has the drawback that the user must hold a platform able to do the recombination, that his work is linear in the number of servers. Also, an external eavesdropper who see all shares transiting through the servers to the user can get the result. In [DK10], a framework is proposed, where the servers cooperate for encrypting the result under a symmetric key (AES) K, and also to encrypt K under the user's public key. The players need not know the plaintext data, neither the secret K. On the receiving side, the user only needs to be able to do public-key and symmetric encryption, which is much more standard than MPC.

As a consequence, research has emerged in the field of symmetric cryptography, to design ciphers that needs a very low number of multiplication, see for instance [DR17], since multiplication is the most delicate operation in MPC (and also in FHE). Nevertheless, there are many situations where AES, being a standard, is required, and it is not possible to use new ciphers not as thoroughly evaluated as AES.

In a similar vein, MPC can used to support distributed cryptographic primitives, like distributed key generation [DM10]. It can also to have a group of servers generate authentication tokens, without knowing the secret data used for building the tokens, for example in a car sharing system [SAM+17, Dho16].

In the following, we assume that variants of these ideas can be deployed for offering computing facilities on encrypted data.

6.2 MPC as a computing network

Secure multiparty computation can be thought as MPC-as-a-service, where one or several users distribute shares of their data to several computing nodes whose role is to do service computation. Thanks to secret sharing, homomorphic properties and MPC, the computing nodes can cooperate to evaluate any function the users require. This has been proposed for example in the Sepia system [BSMD10] where several network operators wish to do event correlation and statistics aggregation. See also [LS11], where it is shown that, in the setting of *secure outsourced computation* a less stringent security model can be used than the standard MPC requirements[44].

In a similar vein, MIT's Enigma [ZNP15] proposal (see also Zyskind's master thesis [Zys06]) is to create a private marketplace, where users can distribute shares of their data to some computing peers, who can, on user's demand, do computation and report the result of computation. Enigma's can also be supported by a blockchain token to give incentives to computing nodes for computing and well behaving.

6.3 MPC as a privacy preserving reporting tool

A possible scenario is when users, e.g. companies, have to interact with analysts, who have to do various studies on users' data, yet the users do not want to reveal all their data to the analysts. In contrast to previous scenario, the users and the analysts are different, but MPC allows users to submit shares to a MPC platform, which reports only the minimum possible information to analysts, see for instance Rmind [BKLS14]. Similarly, Estonia made a prototype for detecting and reporting VAT tax fraud without compromising companies data [BJSV15]. Stanford's Prio is also a recent platform where aggregate statistics over private user data can be computed [CGB17].

6.4 Best available generalistic protocols

6.4.1 Are specific protocols faster ?

Let us take the example of *private set intersection* (PSI). For two parties in the semi honest setting, the recent benchmark [PSZ18] indeed found that "generic circuit-based protocols are less efficient than the newer, OT-based constructions"

But this remark was immediatly followed by: "but [generic approaches] are more flexible and can easily be adapted for computing variants of the set intersection functionality". Indeed, mere algorithms for private-set intersection might face formatting issues, similar to real-life problems with big data. Assume that two entities (bank or hospital) want to match clients that they have in common: the same client might not have exactly the same tag in the two databases. [HHIL+17] provides solutions for two semi-honest participants plus a semi-honest third party (an intermediary with no access to private data). See the ongoing european project [pro17] for a survey on available protocols for secure multiparty big data analytics, (in particular §2.6 on private set intersection).

[44] The general theory of secure multiparty computation enables to consider more complicated settings than an upper bound on the number of adversaries, using the notion of *access structure*, which is simply the set of all the authorized sets and the set of all the adversarial serts. The reference [LS11] shows that "easier" *access structures* can be afforded when MPC is used for outsourced computations

We are not aware of any implementation benchmark for the (recently studied) case of n parties with active adversaries[45]. [GN17] proposes the first UC specific protocol (but assuming a n-parties linear OT), along with an asymptotic study of complexities (in Table 1 loc. cit.). It remains to be benchmarked against generalistic preprocessing-based implementations (for $t < n$ malicious adversaries), in particular [KOS16].

6.4.2 Fastest generalistic implemented protocols

As of February 2018 we can report the following. All timings are from experiments on local networks by the authors. To fix ideas, [GLNP15, table 7] shows that a one-round two party protocol becomes $5 - 10$ times slower when deployed between Ireland and the US. Finally we are not aware if the most recent preprocessing-based protocol [KPR18][46], which also performs 2 and 3 party computations, competes with protocols against active adversaries mentionned below.

- two-party computation (2PC) with security against *passive adversaries*: [GLNP15]. It is based on preprocessed garbled circuits and not proven to be UC. Preprocessing 1000 garbled circuits for AES encryption with the same key takes the authors $0.4s$, then 1000 encryptions (in one round) using these circuits takes $0.2s$.

- 2PC with security against *active adversaries*: [WM17] (based on garbled circuits and not proven to be UC) is fast for one single execution: with statistical security parameter 40, an encryption takes $0.07s$. For a better *amortized cost on many executions and with preprocessing* —and UC security asuming oblivious transfer— then [NST17, Table 4] computes 1024 AES encryptions in a single round (on a local LAN) in $0.008s$, provided a preprocessing of $61s$ —this timing dropping down to $3s$ when tailored to AES (and not generalistic). This paper furthermore provides a benchmark with other protocols[47].

- three-party computation (3PC) with *at most one passive adversary or one active adversary in the client/servers model*: [AFL+16], proven to be UC, is the fastest and is directly designed for arithmetic circuits. Contrary to the previous algorithms it takes more than one round, because every multiplication needs every player to send one field element to other players. However he size of messages being smaller than with garbled circuits, more computations can be done in parallell on the same network. Summarizing: the protocol takes more latency than the previous ones: $0.2s$ for one single AES, whereas the throughput is much more larger: $1,300,000$ AES computations in one second (so 1000 AES in $0.0008s$ if making abstraction of latency). It is unconditionally secure provided correlated randomness assumptions (in practice, correlated randomness is extracted from previous AES computations).

- 3PC with *at most one active adversary*: [FLNW17], implemented and improved in [ABF+17] is based on secret-sharing and preprocessing[48] and is proven to be UC. But the adversary may anonymously abort the protocol and learn the output while the

[45] See [KMP+17] for a recent implementation in the passive case for $n = 5$.

[46] Which is not proven to be UC secure

[47] In particular [RR16], which is also UC secure. On the contrary [LR15], which is not mentionned, is not proven to be uc secure (an ad hoc security notion with indistinguishability is proven) so might be faster. It also remains to be compared to the 2-parties version of [KPR18], though not UC secure

[48] with Beaver's triples and a cut-and-choose scheme

honest parties do not[49] This approach enables more throughput than one-round garbled circuits, at the cost of more latency. Scaling their results, the authors can compute 1000 AES in 0.0005 seconds (making abstraction of the minimum latency of $0.7s$ reported in their table 7). The same table shows that a preprocessing would allow twice more performance.

- $n > 3$ PC with $t < n$ *passive adversaries* [CHK+12] implements the passive version of the classical [GMW] protocol[50] , simulates a private market place that "easily handles tens of customers/providers and thousands of resources successfully". Whereas [BELO16]'s implementation of the classical BMR[51] achieves the same security.

- $n > 3$ PC with $t < n$ *active adversaries*: [DKL+12, Table 1] shows that the online phase of preprocessing-based protocols scales well with the number of parties, since it is essentially non-interactive (from $0.2s$ for a 2PC AES to $0.3s$ for a 10PC AES. By contrast the online phase performance suffers from the number of parties: the UC protocol [KOS16][52] shows that whereas 5000 128 bits multiplications can be preprocessed for 2 parties in one second, only 1000 can be preprocessed for 5 parties. Notice that the recent [KPR18] (relying on hardness of ring LWE) reports 10 times faster preprocessing than the previous for 2 players. It furthermore claims scalable-friendly zero-knowledge proofs of correct encryption, that enable them to implement preprocessing on 100 players while only suffering a division by 5 of performances compared to the 2 players case (Figure 13). All these preprocessing-based protocols do not allow identifiable abort. See subsection 5.3 for (slower) preprocessing-based protocols enhanced with this functionality.

Let us finally mention the Sharemind platform, that featured in 2015 more than 100 different protocols for arithmetic, relational, and database operations with shared values. The authors of [LP14] and [LPJ17] implemented covert security on it, plus [PL15] an automated prover that tests if a protocol leaks information to an active adversary.

References

[ABF+17] Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. Optimized honest-majority mpc for malicious adversaries — breaking the 1 billion-gate per second barrier. *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[AFL+16] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 2016.

[49] Let us also mention [MRZ15] which has the same security properties and, while slower, is easier to implement (based on garbled circuits and standard symmetric cryptography). The preprocessing time for one AES is $0.002s$, while the evaluation time is $0.002s$.

[50] Recall that it is non UC and based on one-way functions, but on the other hand doesn't require preprocessing. GMW also exists in a theoretical version secure against $t < n$ active adversaries and providing identifiable abort

[51] Multiparty garbled circuits, and a priori non UC.

[52] Which also relies on the OT functionality of Figure 4.5 (generalizing [FKOS15] to non binary fields, and thus to long integers)

[AL10] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 2010.

[ALZ13] Gilad Asharov, Yehuda Lindell, and Hila Zarosim. Fair and efficient secure multiparty computation with reputation systems. In *Advances in Cryptology - ASIACRYPT 2013*, 2013.

[AO12] Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In *ASIACRYPT*, 2012.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, 2011.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology — CRYPTO '91*, 1992.

[Bea97] Donald Beaver. Commodity-based cryptography (extended abstract). In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, 1997.

[BELO16] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 2016.

[BGI+14] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *CRYPTO 2014*, 2014.

[BGP92] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. *Bit Optimal Distributed Consensus*. 1992.

[BJSV15] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, 2015.

[BKLS14] Dan Bogdanov, Liina Kamm, Sven Laur, and Ville Sokk. Rmind: a tool for cryptographically secure statistical analysis. *IACR Cryptology ePrint Archive*, 2014:512, 2014.

[BLN16] Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. *Dual EC: A Standardized Back Door*. 2016.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 1983.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, 1990.

[BOO10] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with dishonest majority. In *CRYPTO'10*, 2010.

[BOS16] Carsten Baum, Emmanuela Orsini, and Peter Scholl. Efficient secure multiparty computation with identifiable abort. In *TCC 2016: Theory of Cryptography*, 2016.

[BSFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *CRYPTO*, 2012.

[BSMD10] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas A. Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 223–240. USENIX Association, 2010.

[BTH08] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *Theory of Cryptography*, 2008.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings IEEE International Conference on Cluster Computing*, 2001.

[Can06] Ran Canetti. Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465, 2006.

[CBO14] Ivan Damgård Carsten Baum and Claudio Orlandi. Publicly auditable secure multi-party computation. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, 2014.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 1988.

[CCM08] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008.

[CD09] Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *Advances in Cryptology - CRYPTO 2009*, 2009.

[CD17] Ignacio Cascudo and Bernardo David. Scrape: Scalable randomness attested by public entities. In *Applied Cryptography and Network Security*, 2017.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'00, 2000.

[CDN] Ronald J.F. Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure multiparty computation : an information-theoretic approach.* Cambridge University Press.

[CFY17] Robert Cunningham, Benjamin Fuller, and Sophia Yakoubov. Catching mpc cheaters: Identification and openability. In Junji Shikata, editor, *ICITS 2017*, 2017.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, 2017.

[CGHZ16] Sandro Coretti, Juan Garay, Martin Hirt, and Vassilis Zikas. Constant-round asynchronous multi-party computation based on one-way functions. In *Advances in Cryptology – ASIACRYPT 2016*, 2016.

[CGLR17] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. (leader/randomization/signature)-free byzantine consensus for consortium blockchains. *CoRR*, abs/1702.03068, 2017.

[CGS16] André Chailloux, Gus Gutoski, and Jamie Sikora. Optimal bounds for semi-honest quantum oblivious transfer. *Chicago J. Theor. Comput. Sci.*, 2016.

[CHK+12] Seung Geol Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In *Topics in Cryptology – CT-RSA 2012*, 2012.

[CHOR16] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of secure multiparty computation without broadcast. In *Theory of Cryptography*, 2016.

[CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology – ASIACRYPT*, 2014.

[Cle86] R Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, 1986.

[CW92] Brian A. Coan and Jennifer L. Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Inf. Comput.*, 1992.

[Cyb18] Estonia) Cybernetica, (Talinn. Sharemind platform. https://sharemind.cyber.ee/secure-computing-platform/, 2018.

[DBP14] Sven Laur Dan Bogdanov, Peeter Laud and Pille Pullonen. From input-private to universally composable secure multiparty computation primitives. *Proc. of CSF'14. IEEE Computer Society*, 2014.

[DGH12] Casey Devet, Ian Goldberg, and Nadia Heninger. Optimally robust private information retrieval. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 269–283, 2012.

[DGN10] Ivan Damgård, Martin Geisler, and Jesper Buus Nielsen. From passive to covert security at low cost. In *TCC*, 2010.

[Dho16] Siemen Dhooghe. *Applying Multiparty Computation to Car Access Provision.* PhD thesis, KU Leuven, 2016.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO 2005*, 2005.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, 2006.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Kroigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Advances in Cryptology – EUROCRYPT 2010*, 2010.

[DK10] Ivan Damgård and Marcel Keller. Secure multiparty aes. In *Financial Cryptography and Data Security*, 2010.

[DKL+12] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*, 2012.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the spdz limits. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, 2013.

[DKR06] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*, 2006.

[DM10] Ivan Damgård and Gert Læssœ Mikkelsen. Efficient, robust and constant-round distributed RSA key generation. In Daniele Micciancio, editor, *Theory of Cryptography*, pages 183–200, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology - CRYPTO 2007*, 2007.

[DNPR16] Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael Raskin. On the communication required for unconditionally secure multiplication. In *CRYPTO*, 2016.

[DO10] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In *CRYPTO*, 2010.

[Dow16] Rafael Baião Dowsley. *Cryptography Based on Correlated Data: Foundations and Practice.* PhD thesis, Karlsruher Instituts für Technologie, 2016.

[DR17] Martijn Stam Dragos Rotaru, Nigel P. Smart. Modes of operation suitable for computing on encrypted data. *IACR Transactions on symmetric cryptology*, 2017.

[ELC12] Sayed M. Saghaian Nejad Esfahani, Ying Luo, and Sen-Ching S. Cheung. Privacy protected image denoising with secret shares. In *19th IEEE International Conference on Image Processing, ICIP 2012, Lake Buena Vista, Orlando, FL, USA, September 30 - October 3, 2012*, 2012.

[EMV17] N.P Smart E. Makri, D. Rotaru and F. Vercauteren. Pics: Private image classification with svm. https://eprint.iacr.org/2017/1190.pdf, 2017.

[Fer07] Dan Shumow Niels Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng. CRYPTO 2017 Rump session, 2007.

[FGMvR02] Matthias Fitzi, Nicolas Gisin, Ueli Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology — EUROCRYPT*, 2002.

[FKOS15] Tore Kasper Frederiksen, Marcel Keller, Emmanuela Orsini, and Peter Scholl. A unified approach to mpc with preprocessing using ot. In *Advances in Cryptology – ASIACRYPT 2015*, 2015.

[FLNW17] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *Advances in Cryptology – EUROCRYPT 2017*, 2017.

[FM00] Mattias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, 2000.

[FWW04] Matthias Fitzi, Stefan Wolf, and Jürg Wullschleger. Pseudo-signatures, broadcast, and multi-party computation from correlated randomness. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, 2004.

[GGOR13] Juan Garay, Clint Givens, Rafail Ostrovsky, and Pavel Raykov. Broadcast (and round) efficient verifiable secret sharing. In *ICITS*, 2013.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj M. Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, 2014.

[GIP15] Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In *CRYPTO*, 2015.

[GIW16] Daniel Genkin, Yuval Ishai, and Mor Weiss. Binary amd circuits from secure multiparty computation. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, 2016.

[GLNP15] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[GMRW13] S. Dov Gordon, Tal Malkin, Mike Rosulek, and Hoeteck Wee. Multi-party computation of polynomials and branching programs without simultaneous interaction. In *EUROCRYPT*, pages 575–591, 2013.

[GMW] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, New York, NY, USA.

[GN17] Satrajit S Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. 2017.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, 2006.

[HHIL+17] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. 2017.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *CRYPTO 2011*, 2011.

[HMP00] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *Advances in Cryptology — ASIACRYPT 2000*, 2000.

[IKM+13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *Theory of Cryptography*, 2013.

[IKP+16] Yuval Ishai, Eyal Kushilevitz, Manoj Prabhakaran, Amit Sahai, and Ching-Hua Yu. Secure protocol transformations. In *Advances in Cryptology – CRYPTO 2016*, 2016.

[IOS12] Yuval Ishai, Rafail Ostrovsky, and Hakan Seyalioglu. Identifying cheaters without an honest majority. In *Theory of Cryptography*, 2012.

[IOZG14] Yuval Ishai, Rafail Ostrovsky, Vassilis Zikas, and Rosario Gennaro. Secure multi-party computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2014*, 2014.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, 2008.

[KM15] Vladimir Kolesnikov and Alex J. Malozemoff. Public verifiability in the covert model (almost) for free. In *Advances in Cryptology – ASIACRYPT 2015*, 2015.

[KMP+17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making spdz great again. to appear in Eurocrypt, 2018.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2017.

[LP14] Peeter Laud and Alisa Pankova. Verifiable computation in multiparty protocols with honest majority. In *Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings*, 2014.

[LPJ17] Peeter Laud, Alisa Pankova, and Roman Jagomägis. Preprocessing based verification of multiparty protocols with honest majority. *PoPETs*, 2017.

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[LS11] Jake Loftus and Nigel P. Smart. Secure outsourced computation. In *AFRICACRYPT 2011*, 2011.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 1982.

[LSSV16] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More efficient constant-round multi-party computation from bmr and she. In *TCC*, 2016.

[Mal16] Alexis J. Malozemoff. *EFFICIENT SECURE COMPUTATION FOR REAL-WORLD SETTINGS AND SECURITY MODELS*. PhD thesis, University of Maryland, 2016.

[MBOW88] Shafi Goldwasser Michael Ben-Or and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 1988.

[MRZ15] Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and secure three-party computation: The garbled circuit approach. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[NME13] Isheeta Nargis, Payman Mohassel, and Wayne Eberly. Efficient multiparty computation for arithmetic circuits against a covert majority. In *AFRICACRYPT*, 2013.

[NR17] Jesper Buus Nielsen and Samuel Ranellucci. On the computational overhead of mpc with dishonest majority. In Serge Fehr, editor, *Public-Key Cryptography – PKC 2017*, 2017.

[NST17] Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. Constant round maliciously secure 2pc with function-independent preprocessing using lego. In *NDSS '17. San Diego, CA*, 2017.

[OY16] Satoshi Obana and Maki Yoshida. An efficient construction of non-interactive secure multiparty computation. In *Cryptology and Network Security*, 2016.

[PGFW14] Jason Perry, Debayan Gupta, Joan Feigenbaum, and Rebecca N. Wright. Systematizing secure computation for research and decision support. In *Security and Cryptography for Networks*, 2014.

[PL15] Martin Pettai and Peeter Laud. Automatic proofs of privacy of secure multiparty computation protocols against active adversaries. In *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 75–89, 2015.

[Pol16] Antigoni Polychroniadou. *On the Communication and Round Complexity of Secure Computation*. PhD thesis, Aarhus University, 2016.

[pro17] Horizon 2020 SODA project. Deliverable 2.1. https://www.soda-project.eu/wp-content/uploads/2017/02/SODA-D2.1-WP2-State-of-the-art.pdf, 2017.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Trans. Priv. Secur.*, 2018.

[PW92] Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *STACS 92*, 1992.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, 1989.

[RR16] Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[SAM+17] Iraklis Symeonidis, Abdelrahaman Aly, Mustafa Asan Mustafa, Bart Mennink, Siemen Dhooghe, and Bart Preneel. Sepcar: A secure and privacy-enhancing protocol for car access provision. In *Computer Security – ESORICS 2017*, 2017.

[SF16] Gabriele Spini and Serge Fehr. Cheater detection in spdz multiparty computation. In *Information Theoretic Security*, 2016.

[SJK+17] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[UftFPotE15] UaESMC: Usable and Efficient Secure Multiparty Computation (for 7th Framework Programme of the EC). Deliverable 5.2.3. http://www.usable-security.eu, 2015.

[Unr10] Dominique Unruh. Universally composable quantum multi-party computation. In *Advances in Cryptology – EUROCRYPT 2010*, 2010.

[WLC14] Zhaohong Wang, Ying Luo, and Sen-ching Samson Cheung. Efficient multi-party computation with collusion-deterred secret sharing. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 7401–7405, 2014.

[WM17] Xiao Wang and Jonathan Malozemoff, Alex J.and Katz. Faster secure two-party computation in the single-execution setting. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, 2017.

[WW10] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In Tal Rabin, editor, *CRYPTO*, 2010.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, 1982.

[YO16] Maki Yoshida and Satoshi Obana. On the (in)efficiency of non-interactive secure multiparty computation. In *Information Security and Cryptology - ICISC 2015*, 2016.

[ZNP15] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *CoRR*, 2015.

[Zys06] Guy Zyskind. Efficient secure computation enabled by blockchain technology, 2006.