Secure Distributed Modular Exponentiation: Systematic Analysis and New Results

Bart Mennink

***Abstract*—Modular exponentiation is a vital function in public key cryptography. Dozens of protocols, including encryption schemes, signature schemes, pseudorandom functions, and more, perform this operation on a secret base and/or a secret exponent. In a multiparty computation setting, these secret data might be shared over multiple parties who wish to compute this modular exponentiation in a secure and distributed way. However, whereas typical frameworks for secure multiparty computation based on secret sharing provide a basic tool box of secure distributed computation of, most importantly, randomness generation, addition, and multiplication, the status quo of secure distributed modular exponentiation is unsatisfactory. In this work, we provide a complete and comprehensive overview on existing protocols for perfectly secure distributed exponentiation differing depending on whether the inputs and outcomes are public or shared. We perform a detailed complexity computation of the currently existing protocols, observing that earlier authors have overestimated the complexity of their own protocols, and close the remaining open problem: protocols for secure distributed exponentiation with secret base, secret exponent, and public outcome. We prove that the presented protocol is universally composably secure in the presence of malicious adversaries. We finally exemplify the practical relevance of the new protocol by demonstrating how it can be used for pseudorandom generation and signing.**

***Index Terms*—multiparty computation, secret sharing, exponentiation, classification, constant round.**

I. Introduction

Secure multiparty computation has become a main tool in cryptology, and appears to be a leading approach in secure distributed computation over the internet. It considers a set of n parties to compute a certain function f, in such a way that the parties only learn the predetermined output of f, but nothing else about the input. The idea of multiparty computation is already quite mature: generic solutions for any function f date back to Goldreich et al. [39]. The field has advanced tremendously since then, with core incentive to achieve solutions that are as efficient as possible.

One main approach to multiparty computation is based on threshold homomorphic encryption. The approach roots at the seminal works of Cramer et al. [25] and Damgård and Nielsen [30]. These can be equipped with El Gamal encryption [36], Paillier encryption [52], or even fully homomorphic encryption [15], [16], [38], [59]. An alternative main approach to multiparty computation is based on secret sharing [7], [8]. In this setting, one considers data to be secret shared, i.e., data x is shared over the n parties, denoted $[x]$, in such a way that no party has any knowledge about x, but together they can re-construct x from $[x]$. The approach offers high security: for example, by adopting an unconditionally secure secret sharing scheme such as Shamir's [57], malicious parties cannot recover $[x]$ from x as long as at least a certain predetermined number of the n parties is honest.

A pivotal work in the direction of secret sharing based multiparty computation is that of Damgård et al. [28]. They introduce a wide range of very efficient constant-round protocols for addition, multiplication, inversion, and so on, that are unconditionally secure even against malicious adversaries.

If one tolerates a precomputation phase, where a sufficient pool of "raw" secret shared randomness is generated, the protocols can be sped-up significantly. This was first demonstrated by Bendlin et al. (BDOZ) [9], though it does not scale well to a large number of parties. Damgård et al. [31], [32] presented a general solution, dubbed SPDZ, for an arbitrary number of parties. The protocols are unconditionally secure against malicious adversaries. What distinguishes BDOZ and SPDZ from earlier proposals is that the precomputation can be performed quite efficiently: BDOZ used additively homomorphic encryption whereas SPDZ introduced somewhat homomorphic encryption as a tool to securely generate secret shared random data. Keller et al. [46] introduced MASCOT, a framework for an efficient precomputation phase based on oblivious transfers [53]. Their proposal is based on an earlier work of Frederiksen et al. [34] that focused on binary fields. Precomputation based on oblivious transfers beats the homomorphic encryption based ones in that one does not need zero-knowledge proofs or cut-and-choose mechanisms [22] to guarantee security against malicious adversaries.

These protocols, amongst others, have made multiparty computation practical. Various instantiations/implementations of multiparty computation have already been described in literature [4], [10]–[13], [27], [44]. This line has culminated at two leading open-source multiparty software systems: SCALE-MAMBA [2] and MP-SPDZ [45]. From a more theoretical perspective, these works have also lead to a common practice to assume presence of a "basic tool box" of functions, namely, for creating shares, randomness generation, addition, constant multiplication, full multiplication, and opening shares. This tool box is called an *arithmetic black box* and is used as black box for secure computation of many other functions.

A. Secure Exponentiation

With the basic tools such as constant time secret shared addition and multiplication at hand, the next natural building block is secure distributed modular exponentiation, i.e., the computation of b^e from base b and exponent e in a finite field of prime size q, where at least some of the values b, e, and b^e are secret shared and the remaining ones are public.

Bart Mennink is supported by the Netherlands Organisation for Scientific Research (NWO) under grant VI.Vidi.203.099.

This function has a prominent role in public key cryptography. Since the seminal work of Diffie and Hellman [33], dozens of public key cryptographic protocols have been introduced that perform modular exponentiation, where either the base or the exponent is a secret value. Typical examples include public key encryption such as textbook RSA [55], Elgamal [37], Paillier [52], and Cramer-Shoup [26], digital signature schemes such as RSA [55] and Rabin's signature scheme [54], and distributed PRFs [48]. These examples all testify the need to have a solid and well-understood suite of distributed modular exponentiation protocols.

The earliest results in this direction are due to Damgård et al. [28]. However, their approach was internally based on arithmetic circuits for bit decomposition, a protocol that is known to form a particular performance overhead, in particular on the communication complexity. On the other hand, secure distributed exponentiation protocols *without* bit decomposition are rather scarce and underdeveloped. The work at hand centers around this problem.

B. Existing Protocols

The first protocol for distributed exponentiation without bit decomposition is by Naor et al. [48]. Their protocol considers the case where *only* the exponent is secret shared, and both the base and outcome are public: $c \leftarrow \exp(b, [e])$. Their protocol de facto consists of linearly opening the secret $[e]$ *in the exponent*. The protocol is described in Section III-A. Recently, Grassi et al. [40] reconsidered this problem and expanded the protocol of Naor et al. [48] to security against malicious adversaries. We describe their protocol in Section III-A as well. The complexities of the two protocols are listed in Table I.

The first to analyze the problem of distributed exponentiation without bit decomposition *with secret shared outcome* were Ning and Xu [50], [51]. They described two different protocols for secure distributed exponentiation: one protocol $[c] \leftarrow \exp([b], e)$ for public exponent and one protocol $[c] \leftarrow \exp([b], [e])$ for secret shared exponent (both protocols concern a secret shared base and secret shared outcome). For the case of semi-honest adversaries, where adversaries respect the protocol, their protocols were reasonably efficient, although they are dependent on the length of the exponent. For security against malicious adversaries, that may deviate from the protocol, the authors resorted to cut-and-choose, an approach that yielded a disadvantageous overhead. One might say that the works of Ning and Xu have been surpassed in time, noting that they predate the SPDZ introductions [31], [32]. The protocols are listed in Table I, where we updated the complexities to the modern practice to move randomness to the precomputation phase. We remark that this update in particular reduced the number rounds of $[c] \leftarrow \exp([b], [e])$ from 20 to 11 (semi-honest case) and from 22 to 13 (malicious case), compared to what was originally claimed [50], [51].

Aly et al. [1] improved over Ning and Xu [50], [51]. They first described a perfectly secure and efficient protocol $[c] \leftarrow \exp(b, [e])$ for secret shared exponentiation from public base and secret shared exponent, i.e., the protocol left over by Ning and Xu. Then, they demonstrated how this protocol can be used to improve over the protocols of Ning and Xu, both in the semi-honest as in the malicious setting. The protocols of Aly et al. are formally described in Sections III-B-III-D. (These protocols were originally described for any b, but as we will notice in these sections, some only operate for generators b only.) In these sections, we have also performed a proper analysis of the efficiency of these protocols: it turns out that Aly et al. significantly overestimated the round complexity of some of their protocols. For example, by closer investigation of their protocol for $[c] \leftarrow \exp([b], e)$, it turned out that some seemingly sequential steps were in fact parallelizable, allowing a reduction of the number of rounds from 8 to 5. The observation is explained in more detail in Section III-C. We make a similar observation on the protocol where all inputs and outputs are secret shared, $[c] \leftarrow \exp([b], [e])$, going from 13 to 10 rounds as explained in Section III-D. The observation has been confirmed by the authors of [1]. The (updated) complexities of the protocols by Aly et al. are also listed in Table I. The computation of the number of secure multiplications and openings was missing in [1], and is new in our analysis.

In [58], Smart and Talibi Alaoui considered protocols Multiply-G-P and Multiply-G-S, which can roughly be seen as equivalents of the protocols pss and sss, but then over elliptic curve groups, where the main operation is multiplication (later generalized to pairing groups by Aranha et al. [5]). They use pre-computed multiplication triples to perform the latter protocol.

Related to these results is a benchmark of various multiparty protocols in the SCALE-MAMBA system [2] performed by Aly and Smart [3]. One protocol they consider is a numerical computation of $[c] \leftarrow \exp(2, [e])$, which together with a numerical secure computation of $[c] \leftarrow \log(2, [e])$ allows for the secure composition of any numerical computation of exponentiation with a public or shared base and with a shared exponent. Their protocols and analyses are structurally different than ours: whereas ours focus on *modular* exponentiation, theirs restrict the input e to be from a certain range so that the computations are over the reals and the difficulty shifts to computing the fractional remainder.

C. Completing the Picture

After performing a proper classification of state of the art, with a correct and detailed complexity analysis, we can observe that there are two protocols missing: $c \leftarrow \exp([b], e)$ and $c \leftarrow \exp([b], [e])$. The former of the two is, clearly, pointless: in a finite field of prime size q, one can easily compute b offline from $c = b^e$ and e. The absence of a protocol for the latter type of exponentiation is striking: exponentiation with secret base and secret exponent but public outcome appears in various PRF constructions and digital signature schemes (see also Section I-D). We develop protocols for this remaining function in Section IV, both for semi-honest as for malicious adversaries. The protocols come with a detailed efficiency analysis in Section IV-A, that demonstrates that they are comparable with the other protocols for distributed

exponentiation and operate in constant rounds. The numbers are listed in Table I. An analysis of the protocol in the universal composability framework of Canetti [20] is given in Section IV-B.

D. Application

We give example applications of the exponentiation protocols in Section V. One application is that of secure distributed public key encryption, that relies on protocols for exponentiation of public base b to secret shared exponent $[e]$, both with public and shared outcome. The example is based on earlier examples of Aly et al. [1]. Grassi et al. [40] already described applications of exponentiation with public outcome to symmetric cryptography.

These applications all use secure exponentiation protocols with public base. However, applications are not limited as such: many applications would need secure exponentiation protocols with shared base, too. Consider, for instance, the milestone weak PRF construction of Naor et al. from 1999 [48]: $t = m^k$, where m is the message, k the key, and t the tag, computed in a finite field of prime size q. Naor et al. already considered that k could be secretly shared among servers. We go one step further: by employing our new protocol for $c \leftarrow \exp([b], [e])$, this PRF can be evaluated in a distributed manner without revealing any information about neither $[m]$ nor $[k]$. Our new protocol even proves itself useful if the message would first be hashed, i.e., in the distributed PRF construction of [48], as we explain in Section V-B. In this section, we also demonstrate how the protocol can extend to digital signature schemes.

E. Improved Complexity Against Malicious Adversaries

Malicious adversaries differ from semi-honest adversaries in that they might distribute malformed shares and make the outcome incorrect. The malicious adversary resistant protocols of Section III and IV basically rely on evaluating the semi-honest case two times: once for the target exponent $e/[e]$ and once for a randomized one. As becomes clear from Table I, this practice yields an overhead of more than 100% in the number of rounds, and also the multiplication complexity doubles (on average). An alternative approach is to use message authentication codes (MACs) on the relevant data [31], [32], [49]. A more elegant alternative approach is committed MPC of Frederiksen et al. [35], where each party commits to its shares using additively homomorphic commitment schemes.

F. Outline

We will describe some preliminaries and the multiparty computation model that we consider, in Section II. The protocols of Naor et al. and Grassi et al. on exponentiation with a shared exponent and public base and outcome are given in Section III-A. The three protocols of Aly et al. with secretly shared outcome and at least one of the base and exponent being secretly shared are given in Section III-B-III-D. The protocols for exponentiation with secret shared base and exponent and public outcome are given in Section IV. This section also

Table I: Round, multiplication complexity (number of secure multiplication evaluations), and number of openings of n-party protocols for exponentiation without bit decomposition. For the results of [51], k denotes the cut-and-choose parameter and l the $\log_2$ of the length of the exponent.

	semi-honest			malicious			
operation	rounds	multiplications	openings	rounds	multiplications	openings	reference
$[c] \leftarrow \exp(b, [e])$	3	$n+1$	$2n$	8	$2n+5$	$4n+8$	[1] and Section III-B
$[c] \leftarrow \exp([b], e)$	5	$n+1$	$4n+3$	7	$kn+n+1$	$3kn+4n+3$	[51]
	5	$2n+3$	$4n+3$	10	$4n+11$	$8n+19$	[1] and Section III-C
$[c] \leftarrow \exp([b], [e])$	11	$ln+3l+1$	$4ln+5l+1$	13	$kln+ln+3l+1$	$3kln+4ln+5l+1$	[51]
	10	$3n+6$	$6n+7$	20	$6n+18$	$12n+31$	[1] and Section III-D
$c \leftarrow \exp(b, [e])$	1	0	1	3	1	5	[40], [48] and Section III-A
$c \leftarrow \exp([b], e)$	—	—	—	—	—	—	(pointless)
$c \leftarrow \exp([b], [e])$	7	$2n+4$	$4n+7$	14	$4n+13$	$8n+27$	Section IV

includes the in-depth security and efficiency analysis. We conclude in Section V with a more detailed discussion of some of the possible applications of the introduced protocols. The work is concluded in Section VI.

II. Preliminaries

Let q be a large prime number. We write $\mathbb{Z}_q = \{0, \ldots, q-1\}$, and we let $\mathbb{Z}_q^* = \mathbb{Z}_q \backslash \{0\}$ denote the multiplicative group. We write $p = q - 1$.

A. Multiparty Computation Model

We will describe our protocols for n parties $\mathbb{P} = \{P_1, \ldots, P_n\}$, for some $n \in \mathbb{N}$. Adopting the notation of Grassi et al. [40] and Aly et al. [1] to a large extent, a secret sharing of a value $x \in \mathbb{Z}_q$ is denoted by $[x]_q$, and a secret sharing of a value $x \in \mathbb{Z}_q^*$ is denoted $[x]_{q^*}$. If the respective field is irrelevant for the context, the subscripting is omitted. In this case, we denote the respective set by $\mathbb{F}$.

Our results are described relative to the so-called *arithmetic black box* functionality [30]. This functionality, denoted $\mathcal{F}_{\mathsf{ABB}}$, consists of an ensemble of multiparty protocols that can be evaluated securely in a black box manner. It serves as an ideal functionality that is capable to store secret values over a field $\mathbb{F}$ and that generates outputs upon request. The commands that we consider in this work are described in below Definition II.1.

Definition II.1 (Ideal functionality $\mathcal{F}_{\mathsf{ABB}}$)**.** The functionality $\mathcal{F}_{\mathsf{ABB}}$ is an arithmetic black box for the secure computation of the following functions by a set $\mathbb{P}$ of n parties. Here, each value x that is stored in the functionality is associated with a unique identifier $[x]$ given to all parties.

- $\mathsf{input}(x)$: receive a value $x \in \mathbb{F}$ and store it;
- $\mathsf{share}(x)$: create a share $[x]$ for x;
- $\mathsf{rand}(\mathbb{F})$: sample $r \xleftarrow{\$} \mathbb{F}$ and store $[r]$;
- $\mathsf{add}([x], [y])$: compute $z = x + y$ and store $[z]$;
- $\mathsf{mult}([x], [y])$: compute $z = x \cdot y$ and store $[z]$;
- $\mathsf{listmult}([x_1], \ldots, [x_\ell])$: compute $z = x_1 \cdot \ldots \cdot x_\ell$ and store $[z]$;
- $\mathsf{open}([x])$: send x to all parties.

Here, $\mathbb{F}$ is any finite field.

The functionality allows us to discard of all technicalities of secret sharing and multiparty computation needed to implement this ensemble of functions. It will allow us to describe our protocols and prove them securely using the universal composability (UC) framework [20].

Definition II.2 (UC security)**.** A protocol $\mathcal{P}$ universally composably computes a functionality $\mathcal{F}$ if for all adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ for which no environment $\mathcal{E}$ can non-negligibly distinguish if it is interacting with $\mathcal{A}$ and a list of parties $\mathbb{P}$ running $\mathcal{P}$, or with $\mathcal{S}$ and $\mathbb{P}$ running $\mathcal{F}$.

We refer to [19], [21], [42] for a more detailed treatment of how to formalize universal composable protocols and how to formalize the simulators.

We remark that in our case we will focus on both semi-honest and malicious adversaries. The former follows the entire protocol as expected, but tries to retrieve as much information as possible from the info obtained. The latter may deviate from the protocol at its discretion and learn secret data this way. We always assume that there is a sufficient number of honest parties. This number, depends on the particular instantiation. In addition, in the malicious setting, we aim for security with abort, meaning that the parties are ensured that *if* they receive an output, it is *correct*.

B. Instantiation

In our work, we assume that secret sharing is performed using any linear secret sharing scheme. Assume a value x is shared among n parties, each party holding a share x_i. The value x can be retrieved as $\sum_{i=1}^{n} \alpha_i \cdot x_i$, where the α_i are the linear reconstruction coefficients. One might have adopted a (t, n) threshold scheme, where $t \leq n$ denotes the minimum number of shares needed to restore the value x; our analyses apply, still. The protocol is secure under the assumption that a majority of the parties in $\mathbb{P}$ is honest.

We will not put any restriction on how to actually implement the functionality $\mathcal{F}_{\mathsf{ABB}}$. Nevertheless, for comparison of protocols later on, it appears useful to discuss an example implementation. One way to do so is using the SPDZ protocol of Damgård et al. [29], [31]. SPDZ is a two-stage protocol. In the offline or preprocessing phase of SPDZ, a sufficient amount of random shared data is generated. This data consists of random bits $[x]$ for $x \in \{0, 1\}$, random square pairs $([x], [y])$ with $y = x^2 \bmod q$ for $x, y \in \mathbb{Z}_q$, and random multiplication tuples $([x], [y], [z])$ with $z = x \cdot y \bmod q$ for $x, y, z \in \mathbb{Z}_q$. These randomness can be generated efficiently using, e.g., somewhat homomorphic encryption [29], [31] or using oblivious transfer via the MASCOT protocol [46]. In the online phase, the function add can be done essentially for free without communication cost. In protocols, we will regularly write $+$ and $\cdot$ for addition and multiplication with *public* scalar. Secure multiplication mult of two secret shared inputs takes one round of interaction. It consists of one opening. A protocol for fan-in multiplication $\mathsf{listmult}$ from "ordinary" multiplication and randomness generation was described by Damgård et al. [28], following Bar-Ilan and Beaver [6]. Their protocol takes 5 rounds and $5\ell + 4$ multiplications. However, noting that in SPDZ random tuples are pregenerated, fan-in multiplication $\mathsf{listmult}$ can be performed in 2 rounds of interaction, $\ell + 1$ multiplications, and 2ℓ openings:

- take $\ell + 1$ secret shared random values $[r_0], \ldots, [r_\ell]$, along with precomputed multiplications $[s_i] = \mathsf{mult}([r_{i-1}], [r_i^{-1}])$ for $i = 0, \ldots, \ell$, where the indices wrap modulo $\ell + 1$;
- compute and reveal $[y_i] = \mathsf{mult}([x_i], [s_i])$ for $i = 1, \ldots, \ell$, in 2 rounds;
- compute $y = y_1 \cdot \ldots \cdot y_\ell$ and $\mathsf{listmult}([x_1], \ldots, [x_\ell]) = y \cdot [s_0]$ offline.

This instantiation of $\mathcal{F}_{\mathsf{ABB}}$ yields security with abort, i.e., honest parties know that if they get an output from the protocol it is correct, in case at least one party in $\mathbb{P}$ is honest.

Needless to say, this is only one possible instantiation. Although this will be our reference instantiation when it comes to evaluating the performance of the protocols, different

choices might yield instantiations that are advantageous in certain directions.

III. Existing Protocols for Exponentiation Without Bit-Decomposition

We will recap the most recent existing protocols for secure exponentiation without bit decomposition. The computation of

- b^e from $(b, [e])$ is discussed in Section III-A;
- $[b^e]$ from $(b, [e])$ is discussed in Section III-B;
- $[b^e]$ from $([b], e)$ is discussed in Section III-C;
- $[b^e]$ from $([b], [e])$ is discussed in Section III-D.

The four protocols are dubbed psp, pss, sps, and sss, respectively, where the abbreviation stands for "public/secret base, public/secret exponent, public/secret outcome". Note that, unlike what is suggested by Table I, we treat psp first, the reason being that it predates the other protocols.

A. Public Base and Secret Exponent to Public Outcome

1) Semi-Honest Adversaries: Naor et al. [48] described a very simple protocol for exponentiation to a non-zero secret exponent without bit decomposition, with public outcome. The protocol exploits the Lagrange interpolation multipliers and the fact that $e = \sum_{i=1}^{n} \alpha_i \cdot e_i$, where e_i is the share of e held by party P_i (see also Section II-B). The protocol is described in Protocol 1. It takes 1 round and 0 multiplications. It technically contains 1 opening.

Protocol 1 psp for semi-honest adversaries (Naor et al. [48])

Input: public base $b \in \mathbb{Z}_q$ and secret shared exponent $[e]_{p^*}$
Output: public $b^e \in \mathbb{Z}_q$

1: each party P_i locally computes and broadcasts $c_i \leftarrow b^{\alpha_i \cdot e_i}$
2: $b^e \leftarrow c_1 \cdot \ldots \cdot c_n$
3: return b^e

2) Malicious Adversaries: Grassi et al. [40] likewise considered exponentiation to a non-zero secret exponent without bit decomposition, with public outcome, but then in such a way that it is secure against malicious adversaries. The protocol is described in Protocol 2; it includes a superscript "+" to indicate security against malicious adversaries. It takes 3 rounds and 1 multiplication, noting that lines 1 and 4 can be parallelized once lines 2 and 3 are moved up. It contains 5 openings.

Protocol 2 psp^+ for malicious adversaries (Grassi et al. [40])

Input: public base $b \in \mathbb{Z}_q$ and secret shared exponent $[e]_{p^*}$
Output: public $b^e \in \mathbb{Z}_q$

1: $b^e \leftarrow \mathsf{psp}(b, [e]_{p^*})$
2: $[r]_{p^*} \leftarrow \mathsf{rand}(\mathbb{Z}_p^*)$
3: $[s]_{p^*} \leftarrow \mathsf{mult}([e]_{p^*}, [r]_{p^*})$
4: $b^s \leftarrow \mathsf{psp}(b, [s]_{p^*})$
5: $r \leftarrow \mathsf{open}([r]_{p^*})$
6: verify that $b^s = (b^e)^r$
7: return b^e

Grassi et al. [40] proved this protocol psp^+ to securely implement functionality $\mathcal{F}_{\mathsf{ABB\text{-}psp}}$ in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model, where $\mathcal{F}_{\mathsf{ABB\text{-}psp}}$ is defined as follows:

Definition III.1. The functionality $\mathcal{F}_{\mathsf{ABB\text{-}psp}}$ equals the functionality $\mathcal{F}_{\mathsf{ABB}}$ extended with an ideal function psp that takes as input a public base $b \in \mathbb{Z}_q$ and a secret shared exponent $[e]_{p^*}$, and that computes b^e and sends it to the adversary. If the adversary responds with deliver, the function sends b^e to all parties; if not, it sends $\perp$ to all parties.

B. Public Base and Secret Exponent to Secret Outcome

1) Semi-Honest Adversaries: Aly et al. [1] described an elegant protocol for exponentiation to a non-zero secret exponent without bit decomposition. As that of Grassi et al. [40], the protocol exploits the Lagrange interpolation multipliers and the fact that $e = \sum_{i=1}^{n} \alpha_i \cdot e_i$, where e_i is the share of e held by party P_i (see also Section II-B). The protocol is described in Protocol 3. It takes 3 rounds and $n + 1$ multiplications. It contains $2n$ openings.

Protocol 3 pss for semi-honest adversaries (Aly et al. [1])

Input: public base $b \in \mathbb{Z}_q$ and secret shared exponent $[e]_{p^*}$
Output: secret shared $[b^e]_q$

1: each party P_i locally computes $c_i \leftarrow b^{\alpha_i \cdot e_i}$
2: $[c_i]_q \leftarrow \mathsf{share}(c_i)$ for each i
3: $[b^e]_q \leftarrow \mathsf{listmult}([c_1]_q, \ldots, [c_n]_q)$
4: return $[b^e]_q$

2) Malicious Adversaries: They also described a version secure against malicious adversaries. The protocol is described in Protocol 4; it includes a superscript "+" to indicate security against malicious adversaries. It takes 8 rounds and $2n + 5$ multiplications, noting that lines 1 and 4 can be parallelized once lines 2 and 3 are moved up. It contains $4n + 8$ openings.

Protocol 4 pss^+ for malicious adversaries (Aly et al. [1])

Input: public base $b \in \mathbb{Z}_q$ and secret shared exponent $[e]_{p^*}$
Output: secret shared $[b^e]_q$

1: $[b^e]_q \leftarrow \mathsf{pss}(b, [e]_{p^*})$
2: $[r]_{p^*} \leftarrow \mathsf{rand}(\mathbb{Z}_p^*)$
3: $[s]_{p^*} \leftarrow \mathsf{mult}([e]_{p^*}, [r]_{p^*} - [1]_{p^*})$
4: $[b^s]_q \leftarrow \mathsf{pss}(b, [s]_{p^*})$
5: $[z]_{p^*} \leftarrow [s]_{p^*} + [e]_{p^*}$
6: $z \leftarrow \mathsf{open}([z]_{p^*})$
7: $[y]_q \leftarrow b^{-z} \cdot \mathsf{mult}([b^e]_q, [b^s]_q) - [1]_q$
8: $[r']_{q^*} \leftarrow \mathsf{rand}(\mathbb{Z}_q^*)$
9: $[z]_q \leftarrow \mathsf{mult}([r']_{q^*}, [y]_q)$
10: $z \leftarrow \mathsf{open}([z]_q)$
11: verify that $z = 0$
12: return $[b^e]_q$

The verification steps in pss^+ are comparable to those in the protocol of psp^+, but an additional randomized loop must be incorporated as we require secret shared outcome.

Aly et al. [1] proved this protocol pss^+ to securely implement functionality $\mathcal{F}_{\mathsf{ABB\text{-}pss}}$ in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model, where $\mathcal{F}_{\mathsf{ABB\text{-}pss}}$ is defined as follows:

Definition III.2. The functionality $\mathcal{F}_{\mathsf{ABB\text{-}pss}}$ equals the functionality $\mathcal{F}_{\mathsf{ABB}}$ extended with an ideal function pss that takes as input a public base $b \in \mathbb{Z}_q$ and a secret shared exponent $[e]_{p^*}$, and that returns secret shared $[b^e]_q$.

C. Secret Base and Public Exponent to Secret Outcome

1) Semi-Honest Adversaries: Aly et al. [1] suggested to securely compute $[b^e]$ from $([b], e)$ by combining two evaluations of pss for public base g and a random and secret shared exponent. The protocol is described in Protocol 5. It takes 5 rounds and $2n + 3$ multiplications. Note: the original work claims that this protocol takes 8 rounds, seemingly overlooking that the two evaluations of pss can be performed in parallel (in our counting, we consider these two evaluations of pss to be ran in parallel). It contains $4n + 3$ openings.

Protocol 5 sps for semi-honest adversaries (Aly et al. [1])

Input: secret shared base $[b]_{q^*}$ and public exponent $e \in \mathbb{Z}_p^*$
Output: secret shared $[b^e]_{q^*}$

1: $[r]_{p^*} \leftarrow \mathsf{rand}(\mathbb{Z}_p^*)$
2: $[s]_{p^*} \leftarrow -e \cdot [r]_{p^*}$
3: $[c]_{q^*} \leftarrow \mathsf{pss}(g, [r]_{p^*})$
4: $[d]_{q^*} \leftarrow \mathsf{pss}(g, [s]_{p^*})$
5: $[f]_{q^*} \leftarrow \mathsf{mult}([b]_{q^*}, [c]_{q^*})$ ▷ Note that f satisfies $f^e = (bc)^e = b^e \cdot g^{re} = b^e \cdot g^{-s} = b^e \cdot d^{-1}$
6: $f \leftarrow \mathsf{open}([f]_{q^*})$
7: $h \leftarrow f^e$
8: $[b^e]_{q^*} \leftarrow h \cdot [d]_{q^*}$
9: return $[b^e]_{q^*}$

The original protocol was described for any $b \in \mathbb{Z}_q$, but in this case, opening $[f]_q$ in line 6 would reveal whether or not $b = 0$. In fact, the protocol only works for generators b.

2) Malicious Adversaries: The protocol can be made secure against malicious adversaries by eschewing pss in favor of pss^+. We refer to the resulting protocol as sps^+. The protocol takes 10 rounds and $4n+11$ multiplications. It contains $8n+19$ openings.

D. Secret Base and Secret Exponent to Secret Outcome

1) Semi-Honest Adversaries: Aly et al. [1] finally transformed the protocol of sps to one to securely compute $[b^e]$ from $([b], [e])$ by two additional calls to mult and one additional one to pss, all to account for the secrecy of e. The protocol is described in Protocol 6. It takes 10 rounds and $3n + 6$ multiplications. Note: the original work claims that this protocol takes 13 rounds, again seemingly overlooking that the two evaluations of pss can be performed in parallel (in our counting, we consider these two evaluations of pss to be ran in parallel). It contains $6n + 7$ openings.

Just like for sps, the protocol was originally described for $b \in \mathbb{Z}_q$ but it only works if b is a generator.

Protocol 6 sss for semi-honest adversaries (Aly et al. [1])

Input: secret shared base $[b]_{q^*}$ and secret shared exponent $[e]_{p^*}$
Output: secret shared $[b^e]_{q^*}$

1: $[r]_{p^*} \leftarrow \mathsf{rand}(\mathbb{Z}_p^*)$
2: $[s]_{p^*} \leftarrow -\mathsf{mult}([e]_{p^*}, [r]_{p^*})$
3: $[c]_{q^*} \leftarrow \mathsf{pss}(g, [r]_{p^*})$
4: $[d]_{q^*} \leftarrow \mathsf{pss}(g, [s]_{p^*})$
5: $[f]_{q^*} \leftarrow \mathsf{mult}([b]_{q^*}, [c]_{q^*})$ ▷ Note that f satisfies $f^e = (bc)^e = b^e \cdot g^{re} = b^e \cdot g^{-s} = b^e \cdot d^{-1}$
6: $f \leftarrow \mathsf{open}([f]_{q^*})$
7: $[h]_{q^*} \leftarrow \mathsf{pss}(f, [e]_{p^*})$
8: $[b^e]_{q^*} \leftarrow \mathsf{mult}([h]_{q^*}, [d]_{q^*})$
9: return $[b^e]_{q^*}$

2) Malicious Adversaries: The protocol can be made secure against malicious adversaries by eschewing pss in favor of pss^+. We refer to the resulting protocol as sss^+. The protocol takes 20 rounds and $6n+18$ multiplications. It contains $12n+31$ openings.

IV. Secret Base and Secret Exponent to Public Outcome

The discussion of state of the art in Section III misses two variants. One of these variants, the secure computation of b^e from $([b], e)$, is pointless: one can retrieve b from b^e and e. The remaining one, the secure computation of b^e from $([b], [e])$, is meaningful but still non-existent. We describe such a protocol in this section, both for the case of semi-honest as for malicious adversaries.

1) Semi-Honest Adversaries: The protocols is called ssp, where ssp stands for "secret base, secret exponent, public outcome". The protocol is described in Protocol 7, and it is depicted in Figure 1.

Protocol 7 ssp for semi-honest adversaries

Input: secret shared base $[b]_{q^*}$ and secret shared exponent $[e]_{p^*}$
Output: public $b^e \in \mathbb{Z}_{q^*}$

1: $[r]_{p^*} \leftarrow \mathsf{rand}(\mathbb{Z}_p^*)$
2: $[s]_{p^*} \leftarrow -\mathsf{mult}([e]_{p^*}, [r]_{p^*})$
3: $[c]_{q^*} \leftarrow \mathsf{pss}(g, [r]_{p^*})$
4: $[d]_{q^*} \leftarrow \mathsf{pss}(g, [s]_{p^*})$
5: $[f]_{q^*} \leftarrow \mathsf{mult}([b]_{q^*}, [c]_{q^*})$ ▷ Note that f satisfies $f^e = (bc)^e = b^e \cdot g^{re} = b^e \cdot g^{-s} = b^e \cdot d^{-1}$
6: $d \leftarrow \mathsf{open}([d]_{q^*})$
7: $f \leftarrow \mathsf{open}([f]_{q^*})$
8: $h \leftarrow \mathsf{psp}(f, [e]_{p^*})$
9: $b^e \leftarrow h \cdot d$
10: return b^e

The protocol differs from that of Protocol 5 at three main points. In line 2 secure multiplication needs to be performed as the exponent e is secret shared. Likewise, for the computation of h in line 8 the protocol psp must be invoked. On the other hand, in line 6 we open d: after all, the outcome b^e is allowed

to be public, and be opening d, this allows to compute b^e in line 9 more efficiently.

2) Malicious Adversaries: The protocol can be made secure against malicious adversaries by eschewing psp in favor of psp^+, and pss in favor of pss^+. We refer to the resulting protocol as ssp^+. We discuss the efficiency of the protocols ssp and ssp^+ in Section IV-A. Security is treated in Section IV-B.

A. Efficiency

The protocol ssp of Protocol 7 takes 7 rounds and $2n+4$ multiplications. It contains $4n+7$ openings. Here, we note that:

- lines 3 and 4 can be evaluated in parallel and take 3 rounds, $2n+2$ multiplications, and $4n$ openings in total (see Section III-B);
- lines 6 and 7 can be evaluated in parallel and take 1 round, 0 multiplications, and 2 openings in total;
- lines 2 and 5 both take 1 round, 2 multiplications, and 2 openings;
- line 8 takes 1 round, 2 multiplications, and 1 opening;
- the remaining lines are either covered by precomputation or are to be evaluated offline.

Likewise, the cost of sps^+ is 14 rounds and $4n+13$ multiplications. It contains $8n+27$ openings. Here, the re-computation is performed, keeping in mind that one should count the costs of psp^+ and pss^+, instead of psp and pss.

B. Security

We will focus on the security of ssp^+. Recall from Section III-A and Section III-B that protocols psp^+ and pss^+ have been proven to securely implement functionalities $\mathcal{F}_{\mathsf{ABB\text{-}psp}}$ and $\mathcal{F}_{\mathsf{ABB\text{-}pss}}$, respectively, in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model. For brevity, we define

$$\mathcal{F}_{\mathsf{ABB\text{-}psx}} \tag{1}$$

to be the merger of the two functionalities $\mathcal{F}_{\mathsf{ABB\text{-}pss}}$ and $\mathcal{F}_{\mathsf{ABB\text{-}psp}}$ of Definitions III.1 and III.2.

We will argue security of ssp^+ in the $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$-hybrid model. To do so, we first describe the extended functionality $\mathcal{F}_{\mathsf{ABB\text{-}psx\text{-}ssp}}$ below.

Definition IV.1. The functionality $\mathcal{F}_{\mathsf{ABB\text{-}psx\text{-}ssp}}$ equals the functionality $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$ extended with an ideal function ssp that takes as input a secret shared base $[b]_q$ and a secret shared exponent $[e]_{p^*}$, and that computes b^e and sends it to the adversary. If the adversary responds with deliver, the function sends b^e to all parties; if not, it sends $\perp$ to all parties.

We are now ready to prove UC security of ssp^+ against malicious adversaries.

Theorem IV.2. *In the $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$-hybrid model, the protocol ssp^+ securely implements $\mathcal{F}_{\mathsf{ABB\text{-}psx\text{-}ssp}}$ against static malicious adversaries that corrupt up to $n-1$ parties.*

Proof. We have to construct a simulator $\mathcal{S}_{\mathsf{ssp}}$ such that no environment $\mathcal{E}$ can non-negligibly distinguish the real world $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$ composed with ssp^+ from the ideal world $\mathcal{F}_{\mathsf{ABB\text{-}psx\text{-}ssp}}$ composed with $\mathcal{S}_{\mathsf{ssp}}$. However, as protocol ssp^+ is entirely composed of functionalities from $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$, the description of such simulator is trivial: it simply uses those to simulate ssp^+ indistinguishably, and relays a negative deliver, if any. The protocol ssp^+ thus inherits the security from the operations in $\mathcal{F}_{\mathsf{ABB\text{-}psx}}$. □

V. Application

The protocols of secure distributed exponentiation of Section III and Section IV serve a general purpose and can be used in many settings and applications.

A. Application of Existing Protocols

A typical example, that was also highlighted by Aly et al. [1] is public key encryption. For example, El Gamal encryption [36] is defined over a finite field of prime q elements, generated by element g. It has as public key a field element h, and as secret key an exponent x such that $h = g^x \bmod q$. It operates as follows:

- Encryption of a message m takes a random r and outputs
$$(c_1, c_2) = (g^r \bmod q\,,\; m \cdot h^r \bmod q);$$
- Decryption of a ciphertext (c_1, c_2) outputs $m = c_2 \cdot c_1^{-x}$.

In a distributed environment, one considers a shared input $[m]$ to be encrypted. The ciphertext need not be shared, but in this case, the randomness $[r]$ has to be shared (otherwise, one can recover m from c_2 and h^r). Encryption of this form can thus be evaluated in a distributed manner as

$$(c_1, c_2) \leftarrow \Big(\mathsf{psp}(g, [r]_{p^*})\,,\; \mathsf{open}\Big(\mathsf{mult}\big([m]_q, \mathsf{pss}(h, [r]_{p^*})\big)\Big)\Big)\,.$$

Decryption, likewise, can be done as

$$[m]_q \leftarrow c_2 \cdot \mathsf{pss}(c_1, -[x]_{p^*})\,.$$

The application to other public key encryption schemes that rely on exponentiation is identical. Further immediate applications of protocols with public base appear in the direction of share conversion [24], and in the context hashing based on hardness assumptions, e.g., in VSH [23].

B. Application of New Protocols

These applications only give a modest impression of what can be achieved with the protocols of this work: remarkably, many use cases of exponentiation in public key cryptography apply to a setting where both the base and the exponent are supposed to be kept secret, and the outcome can be public. Here, the newly introduced protocol ssp, where a secret base is exponentiated with a secret value to obtain a public outcome, comes into place.

A notable application of this particular protocol is the landmark distributed PRF of Naor et al. from 1999 [48], later marked as key homomorphic PRF by Boneh et al. [14]. It is defined over a finite field of prime q elements, and on input of a key $k \in \mathbb{Z}_p^*$ and message $m \in \mathbb{Z}_q^*$, it computes a tag as

$$t = m^k \bmod q\,.$$

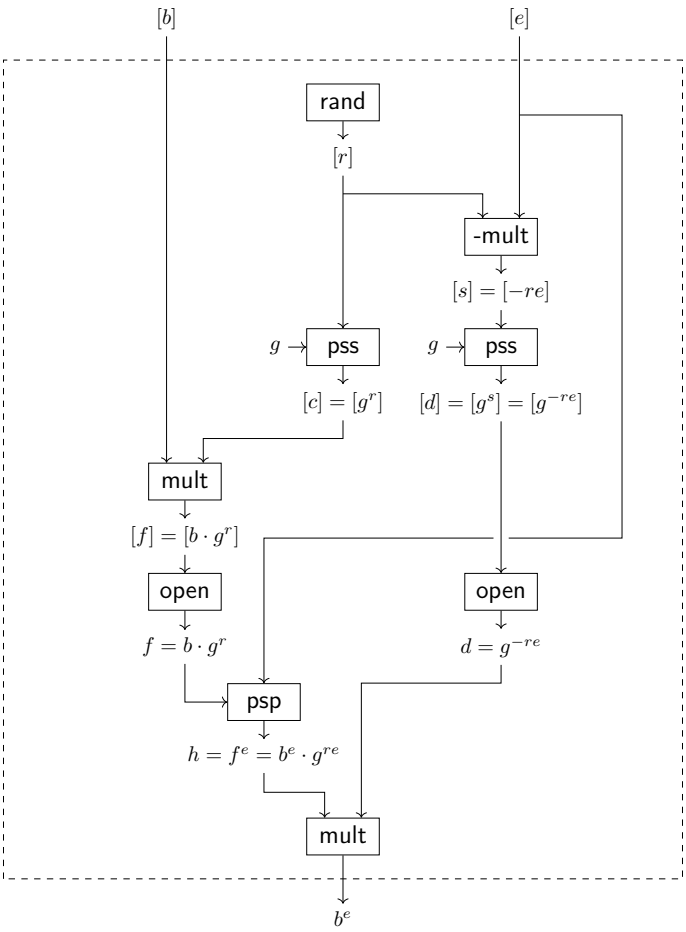

Figure 1: Simplified depiction of ssp of Protocol 7 for semi-honest adversaries.

Naor et al. already considered distributed computation with respect to the key, but not with respect to the message. In a distributed environment, where both inputs $[m]$ and $[k]$ are shared, the parties can evaluate the PRF in a distributed manner as simple as

$$t \leftarrow \mathsf{ssp}([m]_{q^*}, [k]_{p^*}) .$$

Note that there is no reason to keep the tag t shared, it can be made publicly available. We further remark that more modern variants of the PRF of Naor et al. first hash the message, and only authenticate the resulting value:

$$t = H(m)^k \bmod q .$$

In this case, *if the message m has sufficient entropy and if H is assumed to be a random oracle*, it might be sufficient to resort to protocol psp evaluated on public $H(m)$ and shared $[k]$. If m would not have enough entropy, it might be derivable from $H(m)$ by exhaustive search, and it is advisable to use ssp regardless. (We remark that protecting the main application in case of passwords with low entropy is beyond the scope of this work. We refer the reader to [17], [18], [43], [61] for more details.)

Likewise, minor adjustments can be made to extend the technique to rings. In this case, they become relevant, for example, for the RSA signature scheme [55]. Let p, q be two primes. Set $n = p \cdot q$, and let $\varphi(n) = (p-1)(q-1)$ be Euler totient function applied to n. Pick an appropriate value $e \leq \varphi(n)$ such that $\gcd(e, \varphi(n))$, and set $d = e^{-1} \bmod \varphi(n)$. Set (n, e) as public key and d as private key ($p, q, \varphi(n)$ can be discarded). An RSA signature on a message m is computed as

$$\sigma = H(m)^d \bmod n ,$$

where H is a cryptographic hash function as above. Verification of a message-signature tuple (m, σ) is done by checking whether the following holds:

$$\sigma^e \stackrel{?}{=} H(m) \bmod n .$$

Similar methodology as for the PRF of Naor et al. applies. In detail, in a distributed environment, where both inputs $[m]$ and $[d]$ are shared, the parties can sign m in a distributed manner as follows:

$$\sigma \leftarrow \mathsf{ssp}([H(m)_n], [d]_{\varphi(n)}) .$$

VI. Conclusion

We performed an extensive analysis of protocols for perfectly secure distributed modular exponentiation, covering all meaningful cases of secret base and/or secret exponent and/or

secret outcome. We observed that earlier authors have overestimated the complexity of their own protocols, and closed the remaining open problem of exponentiation with secret base, secret exponent, and public outcome.

The comparison of the protocols has mostly been done at a theoretical level, counting the total number of multiplications and openings. An implementation of the protocols in a multiparty software systems such as SCALE-MAMBA [2] or MP-SPDZ [45] could be of value. In addition, although we have done our best to optimize the protocol, we could not formally prove that the schemes cannot be significantly optimized further.

References

[1] Aly, A., Abidin, A., Nikova, S.: Practically Efficient Secure Distributed Exponentiation Without Bit-Decomposition. In: Meiklejohn, S., Sako, K. (eds.) Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10957, pp. 291–309. Springer (2018)

[2] Aly, A., Cong, K., Cozzo, D., Keller, M., Orsini, E., Rotaru, D., Scherer, O., Scholl, P., Smart, N.P., Tanguy, T., Wood, T.: SCALE–MAMBA v1.9: Documentation (July 2020)

[3] Aly, A., Smart, N.P.: Benchmarking Privacy Preserving Scientific Operations. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11464, pp. 509–529. Springer (2019)

[4] Aly, A., Vyve, M.V.: Practically Efficient Secure Single-Commodity Multi-market Auctions. In: Grossklags and Preneel [41], pp. 110–129

[5] Aranha, D.F., Dalskov, A.P.K., Escudero, D., Orlandi, C.: Improved Threshold Signatures, Proactive Secret Sharing, and Input Certification from LSS Isomorphisms. In: Longa, P., Ràfols, C. (eds.) Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12912, pp. 382–404. Springer (2021)

[6] Bar-Ilan, J., Beaver, D.: Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. In: Rudnicki, P. (ed.) Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989. pp. 201–209. ACM (1989)

[7] Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 420–432. Springer (1991)

[8] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10. ACM (1988)

[9] Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic Encryption and Multiparty Computation. In: Paterson, K.G. (ed.) Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6632, pp. 169–188. Springer (2011)

[10] Bogdanov, D., Jõemets, M., Siim, S., Vaht, M.: How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation. In: Böhme, R., Okamoto, T. (eds.) Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8975, pp. 227–234. Springer (2015)

[11] Bogdanov, D., Kamm, L., Kubo, B., Rebane, R., Sokk, V., Talviste, R.: Students and Taxes: a Privacy-Preserving Study Using Secure Computation. PoPETs 2016(3), 117–135 (2016)

[12] Bogdanov, D., Kamm, L., Laur, S., Sokk, V.: Rmind: A Tool for Cryptographically Secure Statistical Analysis. IEEE Trans. Dependable Sec. Comput. 15(3), 481–495 (2018)

[13] Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T.P., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M.I., Toft, T.: Secure Multiparty Computation Goes Live. In: Dingledine, R., Golle, P. (eds.) Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers. Lecture Notes in Computer Science, vol. 5628, pp. 325–343. Springer (2009)

[14] Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key Homomorphic PRFs and Their Applications. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 410–428. Springer (2013)

[15] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully Homomorphic Encryption without Bootstrapping. Electronic Colloquium on Computational Complexity (ECCC) 18, 111 (2011)

[16] Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. SIAM J. Comput. 43(2), 831–871 (2014)

[17] Brost, J., Egger, C., Lai, R.W.F., Schmid, F., Schröder, D., Zoppelt, M.: Threshold Password-Hardened Encryption Services. In: Ligatti et al. [47], pp. 409–424

[18] Camenisch, J., Lehmann, A., Neven, G.: Optimal Distributed Password Verification. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015. pp. 182–194. ACM (2015)

[19] Camenisch, J., Lysyanskaya, A., Neven, G.: Practical yet universally composable two-server password-authenticated secret sharing. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012. pp. 525–536. ACM (2012)

[20] Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA. pp. 136–145. IEEE Computer Society (2001)

[21] Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 19–40. Springer (2001)

[22] Chaum, D.: Blind Signature System. In: Chaum, D. (ed.) Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983. p. 153. Plenum Press, New York (1983)

[23] Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an Efficient and Provable Collision-Resistant Hash Function. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 165–182. Springer (2006)

[24] Cramer, R., Damgård, I., Ishai, Y.: Share Conversion, Pseudorandom Secret-Sharing and Applications to Secure Computation. In: Kilian, J. (ed.) Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3378, pp. 342–362. Springer (2005)

[25] Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 280–299. Springer (2001)

[26] Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1462, pp. 13–25. Springer (1998)

[27] Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential Benchmarking Based on Multiparty Computation. In: Grossklags and Preneel [41], pp. 169–187

[28] Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3876, pp. 285–304. Springer (2006)

[29] Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical Covertly Secure MPC for Dishonest Majority - Or: Breaking the SPDZ Limits. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8134, pp. 1–18. Springer (2013)

[30] Damgård, I., Nielsen, J.B.: Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 247–264. Springer (2003)

[31] Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty Computation from Somewhat Homomorphic Encryption. In: Safavi-Naini and Canetti [56], pp. 643–662

[32] Damgård, I., Zakarias, S.: Constant-Overhead Secure Computation of Boolean Circuits using Preprocessing. In: Sahai, A. (ed.) Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7785, pp. 621–641. Springer (2013)

[33] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory 22(6), 644–654 (1976)

[34] Frederiksen, T.K., Keller, M., Orsini, E., Scholl, P.: A Unified Approach to MPC with Preprocessing Using OT. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9452, pp. 711–735. Springer (2015)

[35] Frederiksen, T.K., Pinkas, B., Yanai, A.: Committed MPC - Maliciously Secure Multiparty Computation from Homomorphic Commitments. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10769, pp. 587–619. Springer (2018)

[36] Gamal, T.E.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984)

[37] Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory 31(4), 469–472 (1985)

[38] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 169–178. ACM (2009)

[39] Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987)

[40] Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-Friendly Symmetric Key Primitives. In: Weippl et al. [62], pp. 430–443

[41] Grossklags, J., Preneel, B. (eds.): Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers, Lecture Notes in Computer Science, vol. 9603. Springer (2017)

[42] Jia, C., Wu, S., Wang, D.: Reliable Password Hardening Service with Opt-Out. In: 41st International Symposium on Reliable Distributed Systems, SRDS 2022, Vienna, Austria, September 19-22, 2022. pp. 250–261. IEEE (2022)

[43] Jiang, J., Wang, D., Zhang, G., Chen, Z.: Quantum-Resistant Password-Based Threshold Single-Sign-On Authentication with Updatable Server Private Key. In: Atluri, V., Pietro, R.D., Jensen, C.D., Meng, W. (eds.) Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13555, pp. 295–316. Springer (2022)

[44] Kamm, L., Willemson, J.: Secure floating point arithmetic and private satellite collision analysis. Int. J. Inf. Sec. 14(6), 531–548 (2015)

[45] Keller, M.: MP-SPDZ: A Versatile Framework for Multi-Party Computation. In: Ligatti et al. [47], pp. 1575–1590

[46] Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer. In: Weippl et al. [62], pp. 830–842

[47] Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.): CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. ACM (2020)

[48] Naor, M., Pinkas, B., Reingold, O.: Distributed Pseudo-random Functions and KDCs. In: Stern [60], pp. 327–346

[49] Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A New Approach to Practical Active-Secure Two-Party Computation. In: Safavi-Naini and Canetti [56], pp. 681–700

[50] Ning, C., Xu, Q.: Multiparty Computation for Modulo Reduction without Bit-Decomposition and a Generalization to Bit-Decomposition. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6477, pp. 483–500. Springer (2010)

[51] Ning, C., Xu, Q.: Constant-Rounds, Linear Multi-party Computation for Exponentiation and Modulo Reduction with Perfect Security. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 572–589. Springer (2011)

[52] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern [60], pp. 223–238

[53] Rabin, M.: How to exchange secrets by oblivious transfer. Tech. Rep. TR-81, Aiken Computation Laboratory, Harvard University, Harvard (1981)

[54] Rabin, M.O.: Digitalized Signatures and Public-Key Functions as Intractable as Factorization (January 1979), MIT/LCS/TR-212

[55] Rivest, R.L., Shamir, A., Adleman, L.M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 21(2), 120–126 (1978)

[56] Safavi-Naini, R., Canetti, R. (eds.): Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7417. Springer (2012)

[57] Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)

[58] Smart, N.P., Alaoui, Y.T.: Distributing Any Elliptic Curve Based Protocol. In: Albrecht, M. (ed.) Cryptography and Coding - 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16-18, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11929, pp. 342–366. Springer (2019)

[59] Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6056, pp. 420–443. Springer (2010)

[60] Stern, J. (ed.): Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding, Lecture Notes in Computer Science, vol. 1592. Springer (1999)

[61] Wang, D., Wang, P.: Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound. IEEE Trans. Dependable Secur. Comput. 15(4), 708–722 (2018)

[62] Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.): Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. ACM (2016)

Bart Mennink received the Ph.D. degree titled Provable Security of Cryptographic Hash Functions, in 2013, under the supervision of Prof. B. Preneel and Prof. V. Rijmen. He was an NWO Veni Postdoctoral Researcher with Radboud University, Nijmegen, The Netherlands, and the FWO Postdoctoral Researcher with imec-COSIC, KU Leuven, Leuven, Belgium. He completed the M.Sc. thesis on Encrypted certificate schemes and their security and privacy analysis during a nine-month internship with Philips, Eindhoven, The Netherlands. He is an Associate Professor with the Digital Security Group, Radboud University Nijmegen, Nijmegen, The Netherlands. His current research focus is on authentication and encryption.