Compact and Efficient UC Commitments under Atomic-Exchanges

Ioana Boureanu[1] and Serge Vaudenay[2]

[1] Akamai Technologies Limited, UK
[2] EPFL, Lausanne, Switzerland
icarlson@akamai.com,serge.vaudenay@epfl.ch

Abstract. We devise a multiple (concurrent) commitment scheme operating on large messages. It uses an ideal global setup functionality in a minimalistic way. The commitment phase is non-interactive. It is presented in a modular way so that the internal building blocks could easily be replaced by others and/or isolated during the process of design and implementation. Our optimal instantiation is based on the decisional Diffie-Hellman (DDH) assumption and the (adversarially selected group) Diffie-Hellman knowledge (DHK) assumption which was proposed at CRYPTO 1991. It achieves UC security against static attacks in an efficient way. Indeed, it is computationally cheaper than Lindell's highly efficient UC commitment based on common reference strings and on DDH from EUROCRYPT 2011.

keywords: commitment, universal composability

1 Introduction

A neat way to design a secure cryptographic protocol is to show that, even in adversarial environments, it emulates a target ideal functionality [25,1,3,21], i.e., a functionality modelling the corresponding primitive implemented by the protocol. One formalism that resides on this idea is the well-known framework of Canetti's, i.e., the universal composability (UC) [9]. This model is compelling because it comprises a composability proof, i.e., protocols proven secure in the UC-setting are guaranteed to remain secure if and when composed with themselves and/or other protocols in a parallel or sequential manner. In order to UC-realize any multiparty computation it suffices to UC-realize the functionality of (multiple) commitment [11]. Thus, commitments became an essential asset within UC-security.

Communication Models in UC. In the original UC papers [9], it was assumed that the channels were secure. However, this assumption was consequently [10] dropped; we will henceforth refer to these two models as the *secure-channel UC* and the *insecure-channel UC*, respectively. The latter means that in the case of honest real-world executions, one can imagine man-in-the-middle adversaries mounting attacks. To bypass this issue, most UC-secure constructions assume or intrinsically require authenticated channels. In this paper, we will place some focus onto which protocols of interest achieve UC-security solely if authenticated channels in the insecure-channel UC model are assumed, and which do so without this assumption.

Requirements for UC Commitments. It should be clear that it is not straightforward to UC-realize commitments. Beyond seeking for a protocol that is hiding and binding as in standard lines, we need the following properties. **(A)** Ideal adversaries should be able to commit reliably to values that they (may) ignore at the time at the commit. And, ideal adversaries should be able to open the simulated commitments to whatever value needed later. **(B)** The ideal adversary also needs to extract the message inside any commitment, particularly within those generated by the adversary. Both should be done without rewinding. Damgård *et. al.*, in [16], refer to the former requirement above as *equivocability* and to the latter as *extractability*. In fact, these requirements were first put forward in [11,17], and [16] formalized a scheme that would clearly exhibit these constraints (and meet them when properly implemented). Moreover, such a scheme had already been realized in [2] into a multi-commitment protocol. Nonetheless, authenticated channels are needed if insecure-channel UC model is assumed.

Unrestricted Communication & UC Commitments. Unfortunately, UC commitment cannot be realized in the standard, non-augmented, UC model. One way to achieve this UC-realization is to use setups [11,24,2,23,12,24], i.e., to work in the UC-hybrid model where all participants can interact with an ideal functionality whilst carrying out their part.

Efficiency of UC Commitments & UC Authenticated Channels. At EUROCRYPT 2011, Lindell proposed a highly efficient version of UC commitments, in [24], in the UC common reference string (CRS)-hybrid model, under the DDH assumption. Lindell's scheme required approximately 36 exponentiations for commitment and opening, if security against adaptive corruptions is offered. For protection against static corruptions only, 26 exponentiations are needed. Very recently, in [5], Blazy *et al.* proposed new UC-secure commitment protocols, making the ones by Lindell more efficient. In this line, they need 22 exponentiations in the static-corruption case and 26 exponentiations, in the adaptive corruption case.

Both Lindell's and Blazy's protocols need the extra assumption of authenticated channels, being cast in the insecure-channel UC model; this extra assumption is often the case, even if it is not always clearly stated in the papers. To see this, imagine the following setting. Let a sender S and a receiver R be both honest. Suppose the environment sends an input x to S, who will play the committer on x. Let $\mathcal{A}$ be a MiM adversary that picks x'. Imagine that $\mathcal{A}$ plays a sender session with R, committing on x', and a receiver session with S. At the end of the two openings, the honest receiver sends x' to the environment. The environment outputs 1 if $x = x'$. Clearly, this will happen in the above, real-world execution with a probability $\frac{1}{2}$, but in the ideal world with probability 1. So, if no authentication is assumed, then this MiM creates the setting for two distinguishable, real and ideal worlds. The CRS setup cannot prevent it.

In this line, we propose a solution that bypasses the need for authenticated channels by using an unforgeable primitive. (Our proofs additionally rest on the soundness of a proof-of-knowledge employed in our construction). We need fewer exponentiations than in Lindell's case, and (with authenticated channels) the same number as in Blazy's case. But, with our protocol, 10 of the 22 exponentiations only need to be executed once, (even) in the case of multiple commitments. We use a different setup, yielding more lightweight building blocks, and a non-interactive commitment phase, to achieve UC-security over insecure channels in the presence of static adversaries.

Isolation as a setup assumption. Damgård *et. al.* UC-realized multiple commitments [16] by using a setup assumption that relaxes the tamper-resistant hardware token to a functionality that models the partial isolation of a party, i.e., the restriction of *input and output* communication from that party. Damgård *et. al.* offer in fact a general construction (rather than an instantiated protocol), relying on the following fact: if a functionality of isolated parties is available, then witness indistinguishable proofs of knowledge (WI-PoK) can be realized, which further provide a type of PKI that makes UC multiple commitment possible. (See [20] for details on PoK.) In this general setting, the UC-realization relies on the existence of one-way permutations and dense public key, IND-CPA secure cryptosystems with ciphertexts pseudorandom (which can be considered pretty heavy assumptions). In fact, the functionality of isolated parties had been used before, in order to realize specifically proofs of knowledge [15]. In [15], the authors motivated the isolation as a remedy to the fact that, in the PoK, the prover could run a man-in-the-middle attack between a helper and the verifier (resulting in the latter not being sure that a prover knows the due witness). This setting applies to the UC-insecurity cases as well, where the simulation fails in the case of simple relay attacks. Overall, we do find the idea behind the work in [16] convincing indeed, in that computation made in guaranteed isolation may alleviate fundamental shortcomings in UC simulators.

In [7], Boureanu *et al.* introduced atomic exchanges as a UC setup, being a somewhat similar alternative to the isolated parties of Damgård *et. al.* The atomic exchange functionality has a different formulation to Damgård's isolation primitive. The main differences between the two functionalities can be summarized as follows. 1. The atomic notion requires isolation of a single message exchange, instead of an entire protocol session and it is used thus-wise. 2. If a responder R is releasing a response to an atomic query, then –in between the query and the response– R will have received no *incoming* messages from the environment (or from another party). Yet, R can leak as much as he likes to the environment (or to another party). At the same time, an R isolated à la Damgård *et. al.* would have *both incoming and outgoing* communications blocked. 3. Atomicity implies full isolation on the incoming tape (i.e., there is no bit received by an atomically engaged R on its incoming tape). Isolation à la Damgård *et. al.* can be partial, i.e., an isolated body can leak a fixed amount of bits. Linked to the requirements needed from UC commitments, the work in [7] formalizes input-aware equivocal commitment, which is a primitive given initially outside of the UC framework, encapsulating similar requirements to those above demanded from UC commitments. The authors also construct a single, bit-commitment protocol (i.e., not a multi-commitment and not working but on bits) emulating this primitive and then prove that the protocol is UC-secure if two atomic exchanges are granted and assuming secure channels. In this line, we will extend the work in [7], to multiple group-element commitments without secure channels and generalize the methodology therein. We will therefore employ some of the tools introduced in [7].

To meet the requirements (A)–(B), and achieve extraction and (strong) equivocability, the protocols use to public-private pairs of keys, $(\mathsf{pk}_X, \mathsf{sk}_X)$ and $(\mathsf{pk}_E, \mathsf{sk}_E)$, respectively. So, we use atomic exchanges in a minimalistic way to declare/register the public keys once for all. Then, these keys are used in multiple commitments.

There are cases where isolation in atomic exchanges make practical sense. E.g., by setting up a sharp time bound for the response and assuming that a responder communicating with a third party would necessarily produce a timeout [4]. We could use similar techniques as for distance-bounding [8,22]. Isolation is also real when a biometric passport is being scanned inside an isolated reader, or when a creditcard is being read in an ATM machine. It could also make sense in a voting booth (equipped with a Faraday cage), in an airplane, in a tunnel, etc. We could imagine hardware-oriented solutions such as a cell phone (responder) registering a key in a secure booth (sender) preventing external radio communications. The advantages of atomic exchanges over, e.g., tamper-hardware devices were discussed in [16].

Our Contribution. Our contribution is five-fold.

1. In this line of work, we further fine-tune restricted local computation, using atomic exchanges [7]. We use these exchanges judiciously.
2. We formalize a design-scheme C_{LCOM} that would achieve commitment in the UC setting. This is more precise/specified than the one in [11,16]. The blocks within C_{LCOM} are similar to those in [24], but the decommitment block is less heavy, i.e., ours is a witness indistinguishable proof of knowledge (PoK) and not a zero-knowledge proof of knowledge[3].
3. Linked to the above, we offer a different manner of obtaining extraction and strong equivocability: it is based on the Diffie-Hellman knowledge (DHK) assumption [14].
4. We advance a protocol UC-realizing $\mathcal{F}_{\mathsf{LCOM}}$ if a few atomic exchanges are possible at the setup phase. This protocol enjoys even more efficiency than the one in [24]. It is more concrete and it has a more judicious use of setups than its counterparts in [16]. We also show how to transform it into a protocol with other global setups such as a public directory or a CRS.

[3] In [16], a witness indistinguishable PoK is used to create a "weak PKI" as part of a different block, i.e., the initialization/setup block. Our initialization/setup block herein is also more lightweight than the one in [16].

5. We also bypass the need of assuming authenticated channels (intrinsic to our predecessors [24,5]) by using a signature and a proof of knowledge, whose soundness deters MiM.

Structure. Section 2 introduces the hardness assumptions needed for special instances of our scheme. Section 3 presents atomic exchanges, i.e., the UC setups used herein. A commitment-scheme is put forward in Section 4. We then give the necessary requirements for this scheme to UC-realize (multi-)commitment. Section 5 offers a concrete, efficient protocol that implements the aforementioned compact scheme and UC-realizes commitment, with atomic exchanges used in a limited way. Section 6 details on the efficiency of our protocol(s) by comparison to existing ones. Appendix A discusses how to transform our protocol into one based on a global public-key registration with no further ideal functionality to be used between participants.

2 Hardness Assumptions

Definition 1 (DH Key Generator Gen). *A* DH key *is a tuple $K = (G,q,g)$ such that G is a group, q is a prime dividing the order of G, g is an element of G of order q. A* DH key-generator *is a ppt. algorithm* Gen *producing DH keys K such that $|K| = \mathsf{Poly}(\log q)$ and the operations (i.e., multiplication, comparison, membership checking in the group $\langle g\rangle$ generated by g) over their domain can be computed in time $\mathsf{Poly}(\log q)$. We say that (S,S') is a valid K-*DH pair for g^σ *if $S \in \langle g\rangle$ and $S' = S^\sigma$, where $\sigma \in \mathbb{Z}_q$.*

An example of a DH key is $(\mathbb{Z}_p^*, q, g)$ where p,q are primes and $p = 2q+1$, $g \in \mathsf{QR}(p)$, $g \neq 1$.

In the descriptions below, we use an arbitrary ppt. algorithm $\mathcal{B}$ generating some coins ρ and states state. Such ρ and state will be used as auxiliary inputs to some other algorithms in the security games formalized below.

Definition 2 ($\mathsf{ag\text{-}DDH}_{\mathsf{Gen}}$). *The* $\mathsf{ag\text{-}DDH}_{\mathsf{Gen}}$ *assumption relative to a DH key generator* Gen *states that for any polynomially bounded algorithms $\mathcal{A}$ and $\mathcal{B}$ in the next game, the probability that $b = \overline{b}$ is $\frac{1}{2}$ but something negligible, i.e., $\Pr[b = \overline{b}] - \frac{1}{2}$ is negligible:*

1: $(\rho, \mathsf{state}) := \mathcal{B}(1^\lambda; r_{\mathcal{B}})$
2: $K := \mathsf{Gen}(1^\lambda; \rho)$, $(G,q,g) = K$
3: *pick* $\alpha, \beta, \gamma \in_U \mathbb{Z}_q$
4: $A := g^\alpha$; $B := g^\beta$; $C_0 := g^\gamma$; $C_1 := g^{\alpha\beta}$
5: *pick* $b \in_U \{0,1\}$
6: $\overline{b} := \mathcal{A}(1^\lambda, \mathsf{state}, A, B, C_b; r)$

The probability stands over the random coins $r_{\mathcal{B}}$, r, $b \in_U \{0,1\}$ and $\alpha, \beta, \gamma \in_U \mathbb{Z}_q$. The probability is negligible in terms of $\log q$. The algorithms $\mathcal{A}$ and $\mathcal{B}$ are ppt. in terms of $\log q$.

In the above definition, "ag" stands for "adversarially-chosen group". This is a weaker assumption than the usual DDH assumption [19] (which is supposed to be hard for *all* generated groups).

We adopt the strengthening from [7] of the Diffie-Hellman knowledge (DHK0) assumption [14] (for a summary of the latter, refer to [19]).

Definition 3 ($\mathsf{ag\text{-}DHK0}_{\mathsf{Gen}}$). *The* $\mathsf{ag\text{-}DHK0}_{\mathsf{Gen}}$ *assumption relative to a DH key generator* Gen *states that for any polynomially bounded algorithms $\mathcal{A}$ and $\mathcal{B}$, there must exist a polynomially bounded algorithm $\mathcal{E}$ such that the following experiment yields 1 with negligible probability:*

1: $(\rho, \mathsf{state}) := \mathcal{B}(1^\lambda; r_{\mathcal{B}})$
2: $K := \mathsf{Gen}(1^\lambda; \rho)$, $(G,q,g) = K$
3: *pick* $\sigma \in_U \mathbb{Z}_q$

4: $(S,S') := \mathcal{A}(1^\lambda, \mathsf{state}, g^\sigma; r)$
5: *if* (S,S') *is not a valid K-DH pair for* g^σ*, then return* 0
6: $s := \mathcal{E}(1^\lambda, \mathsf{state}, g^\sigma, r)$
7: *if* $S = g^s$*, then return 0*
8: *return 1*

The probability stands over the random coins $r_{\mathcal{B}}$*,* r *and* $\sigma \in_U \mathbb{Z}_q$*. The probability is negligible in terms of* $\log q$*. The algorithms* $\mathcal{E}$ *and* $\mathcal{B}$ *are ppt. in terms of* $\log q$*.*

This assumption means that whatever the algorithm producing valid DH pairs (S,S') for a random g^σ with σ unknown, this algorithm must know the discrete logarithm s of their components except for some negligible cases.

What distinguishes these assumptions from the mainstream DDH and DHK0 assumptions [19] is that these should hold for all K selected by a $\mathcal{B}$ algorithm (even by a malicious one) and not only for some K which is selected by an honest participant. In fact, when it comes to selecting a DH key without a CRS in a two party protocol, the above assumption must hold for any maliciously selected K (since we ignore a priori which party is honest). Hence, the name we use: DH assumptions in an adversarially-chosen group. The latter assumption is a special case of the DH knowledge assumption required to hold in *any* group, introduced by Dent in [19]. Here, we do not require the assumption to hold in any group but rather in those groups G for which we can produce a seed for Gen.

In the next, for readability purposes, we will often omit the additional-input 1^λ from the inputs of the machines that require it, its presence being implicit.

3 UC Functionalities

3.1 The Atomic Setup Functionality

We start with the setup functionality we are going to use in our construction. This functionality is denoted $\mathcal{F}_{\mathsf{atomic}}$. Let *poly* be a polynomial. The $\mathcal{F}_{\mathsf{atomic}}$ ideal functionality involves some participants called *Caller* (C) and *Responder* (R). It works as follows (upon receipt of the messages below).

$\mathsf{Ready}(C,R,M)$ **message from** R**.** In this message, M denotes the description of the Turing machine run by R and the functionality parses the message, stores (C,R,M), and sends the message $\mathsf{Ready}(C)$ to the ideal adversary. Any other tuple starting with (C,R) is erased.[4]
Note that –by the above– R can resend this command to $\mathcal{F}_{atomic}$, possibly with a different M.

$\mathsf{Cancel}(C)$ **message from** R**.** This counts for an abortion from the atomic session. So, the functionality sends the message $\mathsf{Cancelled}(C)$ to the ideal adversary and any tuple starting with (C,R) is erased.

$\mathsf{Atomic}(R,c)$ **message from** C**.** The functionality verifies the existence of a tuple for the pair (C,R). If there is none, is aborts. Let (C,R,M) be the found tuple. The functionality runs $r = M(c)$ for no more than $poly(|c|)$ steps, then sends $(\mathsf{response}, C, R, r)$ to C and the ideal adversary, and $(\mathsf{challenge\text{-}issued}, C, R, c)$ to R and the ideal adversary. Finally, the tuple is erased.

Our objective is to employ $\mathcal{F}_{\mathsf{atomic}}$ as little as possible. It is actually required only to set up public keys. So, we will use it in a key-setup/key-registration block, which is executed between each pair of participants who want to run a commitment protocol. This kind of block is bound to require a setup functionality. We could, for instance, rely instead on trusted third parties to whom we could register keys and obtain the public key of participants in a reliable way. In what immediately follows we describe the 2-party approach, without such PKI. However, a version based on a public directory is discussed in Appendix A.2.

[4] We note that the Turing machine M is deterministic (or an equivalent one, a probabilistic one but with the necessary random coins hard-coded within).

3.2 The Commitment Functionality

We now continue with the functionality of commitment we would like to UC-realize. The (unusual) Init step denotes a part in which the parties involved register some data (e.g., public-private keys) that would be used in the remainder of the run of the protocol to carry out the final task.

The $\mathcal{F}_{\mathsf{LCOM}}$ ideal functionality works as follows (upon receipt of the messages below). It involves some participants called *Sender* (S) and *Receiver* (R).

Init(R) **message from** S**.** If R and S are already defined, abort. Otherwise, define (store) R and S, send an $[\mathsf{initialized}, R, S]$ message to R and to the ideal adversary.

Commit(sid, m) **message.** If this does not come from S, or S is undefined, or sid is not fresh, abort. Otherwise, store $(\mathsf{sid}, m, \mathsf{sealed})$ and send a $[\mathsf{committed}, \mathsf{sid}]$ message to R and to the ideal adversary.

Open(sid) **message.** If this does not come from S, or S is undefined, or sid is new, abort. Otherwise, retrieve $(\mathsf{sid}, m, \mathsf{state})$. If $\mathsf{state} \neq \mathsf{sealed}$, abort. Otherwise, send an $[\mathsf{open}, \mathsf{sid}, m]$ message to R and to the ideal adversary[5], and replace state by opened in the $(\mathsf{sid}, m, \mathsf{state})$ entry.

We note that the above functionality is cast in the insecure-channel UC model. This is in the sense that the delayed outputs (i.e., having the functionality send the opening messages to the ideal adversary as well) would not be needed in the secure channels UC. However, they are needed in the insecure channel UC, since without them the ideal simulator would have problems simulating a real execution in which both parties are honest[6]. Unfortunately, in some cases [15,16] where insecure-channel UC is the underlying model, this delayed output is omitted (which would mean that the simulation of the honest, real-world case is impossible). However, in these very case, it can easily be fixed, because their settings rely on a step of a key-registration, and –in itself– this offers the means for authentication.

We will eventually UC-realize this functionality. However, we can easily (with a slightly more computationally expensive protocol) cast everything in terms of the standard multi-commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$ (see Appendix A.2); the latter functionality can be seen, for instance, in [11].

Unlike $\mathcal{F}_{\mathsf{MCOM}}$ where there is no inner init-phase included and participants/roles are defined upon Commit, $\mathcal{F}_{\mathsf{LCOM}}$ allows multiple commitments from the same sender S to the same receiver R decided at its inner init-phase. In other words, $\mathcal{F}_{\mathsf{LCOM}}$ allows multiple commitments at a *link* level, i.e., *L*COM. So, to UC-realize $\mathcal{F}_{\mathsf{MCOM}}$ with $\mathcal{F}_{\mathsf{LCOM}}$, we just need to integrate the LCOM Init phase in every Commit with a new S-R link.

4 Compact UC Commitments

4.1 A Compact Scheme for $\mathcal{F}_{\mathsf{LCOM}}$

In Figure 1, we show a design of a UC commitment scheme based on several building blocks linked together.

These blocks are as follows: a parameter-generation procedure KeyGen yielding the secret-public key pairs $(\mathsf{sk}_E, \mathsf{pk}_E)$ and $(\mathsf{sk}_X, \mathsf{pk}_X)$; a Register block emulating key-registration; an unforgeable scheme $\mathsf{Comm}_{\mathsf{pk}_X}$ which is a commitment in standard lines extractable under sk_X; an interactive proof either of the message inside the commitment or of the knowledge of the secret key sk_E. Note

[5] Sending to the ideal adversary is necessary for the simulation in the insecure-channel UC model because commitment protocols send the committed message in clear during opening and the ideal adversary must simulate such protocol when both participants are honest, although he cannot get the message by any other mean.

[6] It is often the case that, in the real-world execution, the committed input is eventually sent in clear, as part of the opening phase. To get a correct ideal world simulation, this delayed output from the ideal functionality is needed.

that $\mathsf{auth}(\cdots)$ is a shorthand to stress that the input are messages to be protected, either by some authenticated channel, or by means of a digital signature, with a key registered like for sk_X. All these will be explained formally in the sequel and an instantiation of each will be given (if not before, then in Section 5). We will show that, under the right assumptions, these methods can be implemented in a manner that is neither too expensive, nor does it involve many (atomic) exchanges.

Informal Explanations about the Scheme. Before everything, the participants generate their public and secret keys, e.g., pk_X and sk_X for S. Note that we do not assume a CRS to retrieve them from and –in general– we do not suppose necessarily the same domain for the keys of S and those of R.

Then, the sender essentially registers his public key pk_X to the receiver (while storing the associated secret key sk_X for himself). The receiver does the same for $(\mathsf{pk}_E, \mathsf{sk}_E)$, respectively. Further, based on some mechanism and on the setup functionality, each demonstrates[7] to the other that they hold the corresponding secret-key counterparts. To achieve this phase, we use the Register block. This phase, involving key generation (i.e., KeyGen) and key registration (i.e., Register), is called the *key-setup*.

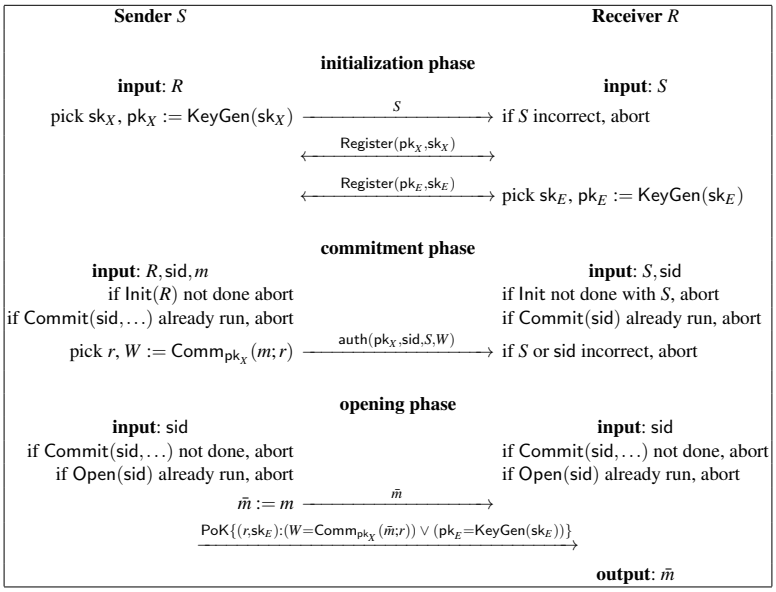

Fig. 1. A Compact Commitment-Scheme C_{LCOM} with Atomic Exchanges

Assume that the sender would like to commit to a message m. Assume that the message is embedded into some suitable domain (e.g., a domain where mathematical operations can be easily applied). The *commitment phase* proceeds as follows. Using his public key pk_X and some random coins r, the sender produces W as the commitment to m using the block Comm. This block is an unforgeable commitment in itself. If it were not unforgeable, we would need to assume authenticated channels (like our predecessors [24,5]), so that a MiM were not able to perturb the honest transactions. I.e., $W = \mathsf{Comm}_X(m;r)$ should be bind S to m and hide m from R. But, to anticipate, if, e.g., an ideal adversary were able to know sk_X for S he could run $\mathsf{Extract}_{\mathsf{sk}_X}(\mathsf{Comm}_{\mathsf{pk}_X}(m;r))$ to obtain m. This would ensure extractability or requirement (B) on page 1.

An essential block of the *opening phase* of this scheme is a *proof of knowledge*, denoted PoK. After sending $\overline{m}$, the sender practically uses this block to prove that either $\overline{m}$ is equal to m and r has been used in producing the commitment, or that he knows sk_E; as only R should know sk_E, this convinces R of

[7] No WI-PoK, as in [16], will be used in this the implementation of this assertion.

the binding character of the commitment. But, obviously, for someone that knows sk_E this commitment becomes equivocal.

Then, for the ideal world to be indistinguishable from the real world, intuitively we need to make sure that the implementation of the blocks are such that their outputs look the same under some coins and an adaptively chosen respective counterpart of those. In the next sections, we will see a way in which this can be achieved.

Note that in order to realize $\mathcal{F}_{\mathsf{LCOM}}$, it is important that the sk and pk keys are fresh for every new pair (S,R) of participants and that Register is run only once for each key.[8]

We proceed with the formalization of these blocks.

4.2 Key Setup Block

We begin by the block of key-setup which includes key generation and key registration. Intuitively, KeyGen computes a public key pk out of a secret key sk. Then, the Register protocol is used for a prover to demonstrate that he holds sk to a verifier who has received pk from this prover. We are going to formalize the semantics of these blocks.

Definition 4 (The KeyGen and Register Blocks). *Let λ be a security parameter. The* KeyGen *block is a function from a domain D_{sk} to a domain D_{pk} (depending on λ). The* Register *block is a ppt. protocol involving a prover P, a verifier V, and an ideal functionality $\mathcal{F}$. The value* sk *is the input for P (which is denoted $P(\mathsf{sk})$). The value* $\mathsf{pk} = \mathsf{KeyGen}(1^\lambda, \mathsf{sk})$ *is the output of V (unless the protocol aborts).*

There must exist a polynomial time algorithm E such that for all ppt. adversary $\mathcal{A}$ and ppt. algorithm $\mathcal{B}$, in an experiment with V, $\mathcal{A}$, and $\mathcal{B}$ having access to $\mathcal{F}$ and V only interacting with $\mathcal{A}$, we have that $\mathsf{KeyGen}(1^\lambda, E(v)) = \mathsf{pk}$*, except with negligible probability, where v denotes the view of $\mathcal{A}$ and* pk *is the output of V.*

For every ppt. algorithm V^* interacting with $P(\mathsf{sk})$, with sk *random, the following happens with negligible probability: V^* outputs s,* $\mathsf{KeyGen}(1^\lambda, s) = \mathsf{KeyGen}(1^\lambda, \mathsf{sk})$*, and P will have not aborted.*

We say that Register *is* authenticating *if there is no man-in-the-middle attack such that a honest verifier ends up with some* pk *such that* $\mathsf{pk} \neq \mathsf{KeyGen}(1^\lambda, \mathsf{sk})$*, where* sk *is the input of the honest prover.*

This non-extractability property is cheaper than zero-knowledge. Note that it implies that KeyGen must be a one-way function.[9] In other words, over a domain $D_{\mathsf{pk}} \times D_{\mathsf{sk}}$ generated as per KeyGen it is computationally hard to retrieve the secret key $\mathsf{sk} \in D_{\mathsf{sk}}$, given the public key $\mathsf{pk} \in D_{\mathsf{pk}}$. In practice, the idea of such a non-extractability of the secret key sk out of the public data pk can rely on the hardness of some computational assumption.

Example 5. We now offer an example of this sort of key-setup. This example is part of the C-at protocol on Fig. 4, page 16. A key-pair $(\mathsf{pk}, \mathsf{sk})$, with pk generated by such an algorithm KeyGen can be given by $((\rho, g^x), (\rho, x))$, i.e., $\mathsf{pk} = (\rho, g^x)$, $\mathsf{sk} = (\rho, x)$, with ρ being some coins to generate $(G, q, g) = \mathsf{Gen}(\rho)$, and where G is a group, q is a prime dividing the order of G, g is an element of G of order q, and $x \in_U \mathbb{Z}_q$. One cannot obtain this sk out of this pk unless they break the $\mathsf{DL}_{\mathsf{Gen}}$ assumption (see Section 2).

We can define Register as follows (see Fig. 2): given $\mathsf{sk} = (\rho, x)$ and $\mathsf{pk} = (\rho, X)$, P sends ρ to V, V computes $(G, q, g) = \mathsf{Gen}(\rho)$, picks $\alpha \in_U \mathbb{Z}_q$, sends an atomic[10] $X_0 = g^\alpha$ to P. Then, P checks $X_0 \in G$

[8] In the C-at protocol to be defined (see Fig. 4), a Register block could be maliciously used as a $z \mapsto z^{\mathsf{sk}}$ oracle, allowing an adversary either to extract or to equivocate a commitment.

[9] One-wayness here means for any ppt. algorithm $\mathcal{A}$ the following probability is negligible in λ: $\Pr_{r_{\mathcal{A}}, \mathsf{sk}}[\mathsf{Gen}(1^\lambda, \mathcal{A}(1^\lambda, \mathsf{pk}; r_{\mathcal{A}})) = \mathsf{pk} \,|\, \mathsf{pk} = \mathsf{Gen}(1^\lambda, \mathsf{sk})]$.

[10] V sending an atomic X_0 is a syntactic-sugar meaning that P sends a prior $\mathsf{Ready}(V, M)$ to $\mathcal{F}_{\mathsf{atomic}}$ where M is an algorithm to compute $M(X_0) = (X, X')$, then V sends $\mathsf{Atomic}(P, X_0)$ to $\mathcal{F}_{\mathsf{atomic}}$. (See [7].)

and sends back X and $X' = X_0^x$ to V. The latter finally checks that $X' = X^\alpha$. Finally, V sends α to P for checking that $X_0 = g^\alpha$.

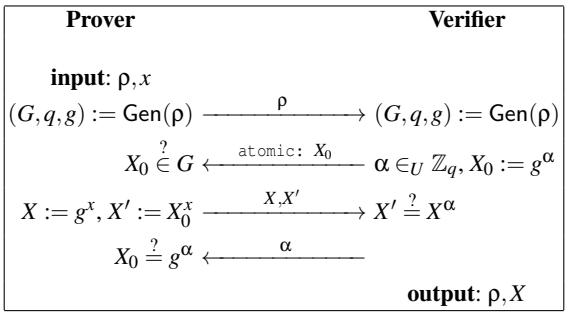

Fig. 2. A Register Protocol

Lemma 6. *Under the* ag-DHK0$_{\mathsf{Gen}}$ *and* DL$_{\mathsf{Gen}}$ *assumptions, the protocol in Example 5 based on* $\mathcal{F}_{\mathsf{atomic}}$ *is a* Register *block with* KeyGen. *It is further authenticating.*

The idea of this protocol is that by preparing the atomic response, the prover provides an algorithm from which we can extract X based on the DHK0 assumption.

Proof. Based on the ag-DHK0$_{\mathsf{Gen}}$ assumption, the atomic response clearly leaks x. So, P's view can provide sk and the first requirement is satisfied.

Furthermore, based on the DL$_{\mathsf{Gen}}$ assumption, the protocol does not leak sk to V^*. This comes from that we could run V^* with a genuine ρ from the DL$_{\mathsf{Gen}}$ game, then continue with some dummy $\bar{X} = g^{\bar{x}}$ and $\bar{X}' = X_0^{\bar{x}}$ to get α (otherwise, P aborts). Then, he rewind to when X and X' are submitted to V^*. He gets a genuine X from the DL$_{\mathsf{Gen}}$ game and sets $X' = X^\alpha$. Clearly, this experiment cannot extract x under the DL$_{\mathsf{Gen}}$ assumption.

The authentication comes from that the atomic functionality authenticates X to the verifier. □

We could also have a Register block based on a global CRS (à la [24]). The prover simply sends $\sigma = \mathsf{Enc}_{\mathsf{crs}}(\mathsf{sk})$ and $\mathsf{PoK}\{\mathsf{sk} : \sigma = \mathsf{Enc}_{\mathsf{crs}}(\mathsf{sk}) \wedge \mathsf{pk} = \mathsf{KeyGen}(\mathsf{sk})\}$.

4.3 The Extractable Commitment Block

We mention the requirements needed from the Comm block (and the Extract block) in the C_{LCOM} scheme; consider the notations therein.

Definition 7 (Extractable Commitment). *An* extractable commitment *for the* KeyGen *and* Register *blocks is defined by a set of algorithms* Comm *and* Extract *such that for all* $\mathsf{sk}_X \in D_{\mathsf{sk}}$*, m, and r, if* $\mathsf{pk}_X = \mathsf{KeyGen}(1^\lambda, \mathsf{sk}_X)$*, then* $\mathsf{Extract}_{\mathsf{sk}_X}(\mathsf{Comm}_{\mathsf{pk}_X}(m; r)) = m$.

Further, we require that an extractable commitment is computationally hiding with the Register *block. I.e., any ppt. algorithm* $\mathcal{A}$ *has a probability of winning the following game which is negligibly close to* $\frac{1}{2}$*:*

1: *pick* $\mathsf{sk}_X \in_U D_{\mathsf{sk}}$ *and set* $\mathsf{pk}_X := \mathsf{KeyGen}(1^\lambda, \mathsf{sk}_X)$
2: *run the* Register *block with* $\mathcal{A}$ *playing the role of the verifier*
3: $\mathcal{A}$ *selects two messages* m_0 *and* m_1
4: *flip a coin b, compute* $W = \mathsf{Comm}_{\mathsf{pk}_X}(m_b; r)$*, and run* $\mathcal{A}$ *on* W
5: $\mathcal{A}$ *outputs* b' *and wins if* $b' = b$

$\mathcal{A}$ may use the functionality $\mathcal{F}$ coming from Register *as per Def. 4.*

The reason why we introduced Register in the hiding notion is because we do not necessarily assume any zero-knowledge property on Register. So, some information may leak, but we want that it does not help to uncover the committed message.

4.4 The Equivocable Opening Block

Definition 8 (PoK **Block).** *Given the blocks* KeyGen *and* Comm *and an instance described by* (W, pk_E) *of an initialization and commitment phase, the* PoK block *is a witness indistinguishable proof of knowledge*[11] *from S to R for either r or* sk_E *such that* $W = \mathsf{Comm}_{\mathsf{pk}_X}(m;r)$ *or* $\mathsf{pk}_E = \mathsf{KeyGen}(\mathsf{sk}_E)$.

By *proof of knowledge*, we mean that the protocol is polynomially bounded, complete, and that there is an extractor who can compute a witness out of the view of a successful malicious prover. By *witness indistinguishable* (WI), we mean that the honest prover can use either r or sk_E as a witness to run his algorithm, and that the respective cases cannot be distinguished by a malicious verifier. (Again, see [20] for details on WI-PoK and PoK.) More concretely, and ppt. algorithm $\mathcal{A}$ has a probability of winning negligibly close to $\frac{1}{2}$ in the following game:

1: $\mathcal{A}$ selects an instance inst and two possible witnesses w_0 and w_1 for PoK
2: flip a coin b and set $\mathsf{wit} = w_b$
3: run PoK with a prover for inst using wit as a witness and with $\mathcal{A}$ playing the role of the verifier
4: $\mathcal{A}$ outputs b' and wins if $b' = b$

$\mathcal{A}$ may use the functionality $\mathcal{F}$ coming from Register as per Def. 4.

4.5 UC Security of the Compact Scheme

Theorem 9. *Under the assumptions of Def. 4 (using a functionality $\mathcal{F}$), Def. 7, and Def. 8, in presence of a static adversary, the compact-scheme C_{LCOM} UC-realizes the $\mathcal{F}_{\mathsf{LCOM}}$ ideal functionality using $\mathcal{F}$ as a global setup.*[12]

In the insecure-channel UC model with authentication, the result holds when $\mathsf{auth}(\cdots)$ is just transmitting messages through the authenticated channel. In the insecure-channel UC model without authentication, the sender must register an additional (authenticated) key and auth simply appends a digital signature based on this key. So, we move the auth requirement to the initialization phase. If the Register block is authenticating, this is solved.

Proof (sketch). Let S (sender) and R (receiver) be two participants running one initialization $S_{\mathsf{init}}/R_{\mathsf{init}}$ and multiple commitments $S_{\mathsf{commit}}/R_{\mathsf{commit}}$ and $S_{\mathsf{open}}/R_{\mathsf{open}}$, upon activation by the environment. Note that S and R are paired by the unique $\mathcal{F}_{\mathsf{LCOM}}$ initialization.[13] In the ideal world, they run, if honest, the dummy_S or dummy_R algorithms forwarding inputs/outputs between the environment and $\mathcal{F}_{\mathsf{LCOM}}$. Otherwise, they behave as instructed by the ideal adversary I. While the ideal-world experiment is running, I runs an internal simulation of the real world experiment to make the interaction with the environment indistinguishable. So, I runs a simulation of the adversary $\mathcal{A}$, of the honest participants S or R supposed to run their specific algorithms, and of the setup functionality $\mathcal{F}$ (in due turns).

[11] See [20] for details on witness indistinguishable proofs of knowledge (WI-PoK).

[12] By *global setup*, we mean that the environment can access to it as well. This is also called GUC in the literature.

[13] So, proving GUC reduces to proving EUC: in a multiparty setting, the participant calling $\mathcal{F}_{\mathsf{LCOM}}$ with the identifier of another participant defines S and R. All other participants can be glued into the environment.

He corrupts correspondingly to the real world the dummy S or R who then behave following the $\mathcal{A}$ simulation.

In what follows, we describe, depending on the corruption state, how the simulation of the honest participants is done. Our simulator will be straight-line, but *proving* (and only proving) that the simulation is indistinguishable may require rewinding, as allowed in the UC model.

Case where S and R are corrupted. There is no honest participant to simulate: $\mathcal{A}$ defines the behavior of S and R and the simulation is perfect. Actually, there is no interaction with $\mathcal{F}_{\mathsf{LCOM}}$ in this case.

Case where S is honest. R may be corrupted or not. If R is honest, its simulation is based on the normal algorithms R_{init}, R_{commit}, and R_{open}. Clearly, this simulation of R to $\mathcal{A}$ is perfect.

During initialization, the simulation of S is straightforward as it requires no communication with the environment: he runs the same algorithms S_{init} as in the real world. This simulation is perfect.

We note that while the honest S is simulated, even though R may be honest as well, his messages may be modified by $\mathcal{A}$. In any case, we consider the honest S interacting with some T where T is the complement of the simulation of S in I. I.e., it includes the simulation of $\mathcal{A}$ and the one of R, no matter whether R is honest or not. Let sk_X be the secret key selected by the simulator of the honest S. Let pk_E be the public key registered to S. Based on the property of the Register block, I can extract sk_E corresponding to pk_E based on the view of T. (In Def. 4, T plays the role of $\mathcal{A}$ while the environment, the dummy honest participants, and $\mathcal{F}_{\mathsf{LCOM}}$ play the role of $\mathcal{B}$.)

During commitment, I simulates S running S_{commit} on some random message m.

During the opening, $\mathcal{F}_{\mathsf{LCOM}}$ tells I the value of $\bar{m}$ committed by the dummy S. Then, I simulates S equivocating the commitment to $\bar{m}$ by using sk_E in the $m \neq \bar{m}$ case: I makes S send $\bar{m}$ and run the PoK protocol with sk_E as a witness. In the $m = \bar{m}$ case, I simulates S normally: using S_{open}.

Indistinguishability. In general, to prove indistinguishability, we have to prove that all messages sent to the environment are indistinguishable in both worlds. There are two types of messages: the output from the dummy (honest) participants (in our case, there is only `dummy_`R, if honest, and during opening, which has content), and the messages from the corrupted ones, i.e., from $\mathcal{A}$. This reduces to proving that `dummy_`R, if honest, opens to a correct message, and that the simulation of honest participants is indistinguishable by $\mathcal{A}$ in both worlds.

Let us consider the honest R case. Clearly, `dummy_`R sends the outcome $\bar{m}$ to the environment, and it matches the input to `dummy_`S. In the real world, even though the adversary may corrupt the communication, we prove that R ending the opening on $\bar{m}$ while S began the commitment with a different message happens with negligible probability. For that, we assume that these messages are different. Thanks to the Register block and `auth` message, both S and R use the same pk_E and W. Since PoK is a sound proof of knowledge, from the prover (i.e., the entire experiment except the simulation for R), we extract a witness, possibly by rewinding. Since the commitment does not open to $\bar{m}$, this witness must be a secret key related to pk_E. Now, since sk_E is only used in Register, this shows that we can extract a preimage of $\mathsf{KeyGen}(\mathsf{sk}_E)$ from the Register protocol. But this is excluded by Def. 4. So, the outcome $\bar{m}$ from a honest `dummy_`R matches the one of the real world experiment.

Then, we have to prove that the simulation of the interaction between S and R (when honest) makes the simulation of $\mathcal{A}$ behave in an indistinguishable way to the adversary in the real world. The case of a honest R is clear: the simulation in the ideal world behaves exactly like in the real world. As for S, the result is clear for $m = \bar{m}$ as they run exactly the same algorithms. It remains to consider the simulation of S in the $m \neq \bar{m}$ case.

Let Γ_0 be the ideal world experiment producing the output of the environment, in the $m \neq \bar{m}$ case. We note that sk_X is only used by Register during the initialization. So, we can use the hiding property

of Comm to say that Γ_0 is indistinguishable to the game Γ_1 in which we run $S_{\mathsf{commit}}(R,\mathsf{sid},\bar{m})$ for S instead of $S_{\mathsf{commit}}(R,\mathsf{sid},m)$. Just as in Γ_0, this game Γ_1 is still using sk_E as a witness to run PoK. Due to the witness indistinguishable property, Γ_1 is indistinguishable to the game Γ_2 in which S uses r as a witness instead. This final game Γ_2 corresponds to the real world experiment. So, the real and ideal world experiments produce indistinguishable outcomes.

Case where S (but not R) is corrupted. During initialization, R is simulated by running the normal algorithm R_{init} interacting with $\mathcal{A}$ and $\mathcal{F}$. So, thanks to the property of the Register block I can extract sk_X based on his own view.

The simulation for the commitment phase starts normally by running the normal algorithm for R. After W is released, I computes $\mathsf{Extract}_{\mathsf{sk}_X}(W)$ to deduce the committed value m by $\mathcal{A}$. If extraction fails, m is set to a random message. Then, the ideal adversary I makes the corrupted `dummy_S` send a $\mathsf{Commit}(\mathsf{sid},m)$ message to $\mathcal{F}_{\mathsf{LCOM}}$.

The simulation for the opening phase starts normally with R running the normal algorithm R_{open}. If R_{open} aborts, I aborts. If it succeeds and R_{open} outputs something, then the ideal adversary I makes `dummy_S` send an $\mathsf{Open}(\mathsf{sid})$ message to $\mathcal{F}_{\mathsf{LCOM}}$.

Indistinguishability. Since R follows his algorithms, the simulation of the interaction (to $\mathcal{A}$) is perfect. We only have to prove that the outcome of `dummy_R` (which will be sent to the environment) matches the one by R. We observe that, due to the extractability of the commitment, it is perfectly binding. So, if R in the real world ends up with the opened commitment $\bar{m}$ and that $\mathsf{Extract}_{\mathsf{sk}_X}(W) \neq \bar{m}$, due to PoK being sound, we could extract (possibly by rewinding) a valid witness sk_E. Since R is honest and only uses sk_E for Register, the properties of Register make it impossible. So, this proves that $\mathsf{Extract}_{\mathsf{sk}_X}(W) \neq \bar{m}$ with negligible probability. So, we have $\bar{m} = m$ in the real world, which is also guaranteed by the simulation. □

5 Instantiated Compact Scheme

Given a group $K = \mathsf{Gen}(1^\lambda,\rho)$, we define an injective function map from the set of possible values to commit to the group K. The function map, as well as its inverse, must be easy to compute. For instance, if $\langle g\rangle$ is the group of quadratic residues in $\mathbb{Z}_p^*$ and $p = 2q+1$ is a strong prime, we can set the message space to $\{1,\ldots,N\}$ for $N < q$ and define $\mathsf{map}(m) = (\pm m) \bmod p$, specifically the only one of the two values which is a quadratic residue.

In Figure 4, on page 16, we present a protocol that implements the schema in Figure 1. Then, we prove that this protocol is UC-secure with atomic as a setup, and under certain assumptions.

The KeyGen and Register blocks are as in Example 5. Based on $\mathsf{pk}_X = (\rho,X)$ and $\mathsf{sk}_X = (\rho,x)$, for $r \in \mathbb{Z}_q$, we have $\mathsf{Comm}_{\mathsf{pk}_X}(m;r) = (U,V)$ with $U = g^r$ and $V = \mathsf{map}(m)X^r$. This is the ElGamal encryption. We let $\mathsf{Extract}_{\mathsf{sk}_X}(U,V) = \mathsf{map}^{-1}(VU^{-x})$.

Lemma 10. *Under the* ag-$\mathsf{DDH}_{\mathsf{Gen}}$ *assumption, the above* Comm *and* Extract *algorithms define an extractable commitment in the sense of Def. 7, for* KeyGen *and* Register *from Example 5.*

Proof. To show that Comm is hiding, we consider the game in Def. 7: the adversary $\mathcal{A}$ receives ρ defining a group with a generator g, then sends some random X_0 in the group, receives X,X', sends α such that $X_0 = g^\alpha$ (otherwise, fail), sends some m_0 and m_1, receives (U,V) which is the ElGamal encryption of $\mathsf{map}(m_b)$ with key X, for some random b, and produces a bit b'. He wins if $b = b'$.

First, we play with $\mathcal{A}$ by submitting some $\bar{X} = g^{\bar{x}}$ for some random $\bar{x}$, with $\bar{X}' = X_0^{\bar{x}}$. Then we can get α and rewind, by submitting some external X and $X' = X^\alpha$. This reduces to the semantic security of the ElGamal encryption. We then use the standard result [6] that ElGamal encryption is IND-CPA secure under the $\mathsf{DDH}_{\mathsf{Gen}}$ assumption. □

By using the standard construction [13] based on proofs of disjunctive statements [13,18], we construct a PoK for our instances. The protocol is depicted on Figure 4 in which the prover uses r as a witness. To use $\mathsf{sk}_E = y$ as a witness (for equivocation), the computations of the prover are replaced by

$$b \in_U \mathbb{Z}^*_{q_2},\ c_1 \in_U \{0,1,\ldots,2^n-1\},\ s_1 \in_U \mathbb{Z}^*_{q_1}$$
$$t_1 := U^{c_1} g_1^{s_1},\ t_2 = (\tfrac{V}{\mathsf{map}(\tilde{m})})^{c_1} X^{s_1},\ t_3 := Y^b$$
$$c_2 := c \oplus c_1,\ s_2 := (b - c_2 y) \mod q_2$$

We have the following result.

Lemma 11. *The 3-move protocol with the t, c, and s messages (in the opening phase) in Figure 4 defines a* Σ*-protocol for* $\{(r,\mathsf{sk}_E) : ((U,V) = \mathsf{Comm}_{\mathsf{pk}_X}(m;r)) \vee (\mathsf{pk}_E = \mathsf{KeyGen}(\mathsf{sk}_E))\}$*. It is a* PoK *block in the sense of Def. 8, for* KeyGen *and* Comm *from above.*

Theorem 9 and Lemma 6–11 wrap up into the following result.

Theorem 12. *Under the* ag-DHK0$_{\mathsf{Gen}}$ *and* ag-DDH$_{\mathsf{Gen}}$ *assumptions, the* C-at *protocol on Figure 4 UC-realizes* $\mathcal{F}_{\mathsf{LCOM}}$ *in the* $\mathcal{F}_{\mathsf{atomic}}$*-hybrid model considered, under a static adversary.*

In Appendix A, we discuss on possible extensions. I.e., relaxing the ag-DHK0$_{\mathsf{Gen}}$ assumption, implementing the atomic exchanges, and making a PKI for multiple commitment.

6 Efficiency

To compare the efficiency of protocols, we count the number of exponentiations. There are some which must be done during the setup and which could be used for several commitments. There are some using small exponents (such as c_1 or c_2) which are faster than others. If we are not satisfied by the DHK0 assumption, we can use the ZK proof based on the DDH assumption as per Appendix A.1 (and if H_{K} is say implemented via Pedersen commitment [27]). We compare the protocols of [24] and [5] with ours below.

protocol	setup	fast	regular
Lindell [24]		6	20
Blazy *et al.* [5]		2	20
our protocol with DHK0	10	4	8
our protocol with DDH	16	4	8

For 2-party protocols requiring many commitments, out protocol is thus at least twice faster than others.

The reduction in the number of exponentiations resides mainly on our use of the ag-DHK0$_{\mathsf{Gen}}$ assumption. As aforementioned, it may be possible to select adversarial groups where the ag-DHK0$_{\mathsf{Gen}}$ assumption may hold and then efficiently work in these groups. An example of this was given in Example 5. Also, to this end, the atomic exchanges are very limited within (and see Appendix A.2 for possible, efficient implementations through, e.g., distance-bounding [22]).

To achieve security, the previous protocols in [24,5] assumed authenticated channels, on top of the insecure-channel UC model. We can relax this assumption by using a signature, at the cost of a few more exponentiations. (E.g., 3 more regular ones for signature and verification, and 5 more during setup for registering verification key.)

All in all, in general, we yield a generally more efficient, very modular UC commitment protocol.

7 Conclusions

In this paper, we devised a design-scheme for multiple (concurrent) commitment-scheme operating on large messages. It uses the ideal setup functionality of atomic messages in a minimalistic way. We suggest how this functionality can be achieved in practice, and we claim that it is indeed lighter than other UC setups for commitments. Our scheme enjoys UC security under static attacks. It is presented in a modular way so that the internal building blocks could easily be replaced by others and/or isolated during the process of design and implementation. Our optimal proposed instantiation is based on the decisional Diffie-Hellman assumption and the adversarially selected group Diffie-Hellman knowledge assumption. This outperforms other efficient UC commitments [24] based on CRS and DDH. At the same time, it can be viewed as an alternative to the new protocol in [5], bypassing the need for authenticated channels, but keeping in place the same number of exponentiations with a more modular construction. However, our protocol can enjoy UC security *without needing to assume authentication* on top of the UC insecure channels, unlike [5,24]. If the adversarially selected group Diffie-Hellman knowledge assumption is dropped, another instantiation of ours performs still slightly better than existent efficient UC commitments.

References

1. M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In *Proc. of TCC 2004*, pages 336–354, Cambridge, MA, USA, 2004.
2. B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. of FOCS 2004*, pages 186–195, Washington, DC, USA, 2004. IEEE Computer Society.
3. D. Beaver. Foundations of secure interactive computing. In *Proc. of CRYPTO 1991*, pages 377–391, London, UK, 1992. Springer-Verlag.
4. T. Beth and Y. Desmedt. Identification tokens or: Solving the chess grandmaster problem. In *Proceedings of CRYPTO 1990*, Lecture Notes in Computer Science, pages 169–176. Springer, 1991.
5. O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In *ACNS 2013*, pages 534–551, 2013.
6. D. Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the Third Algorithmic Number Theory Symposium ANTS 1998*, pages 48–63. Springer-Verlag, 1998.
7. I. Boureanu and S. Vaudenay. Input-Aware Equivocable Commitments and UC-secure Commitments With Atomic Exchanges. PROVSEC 2013, to appear.
8. S. Brands and D. Chaum. Distance-Bounding Protocols (Extended Abstract). In *EUROCRYPT*, pages 344–359, 1993.
9. R. Canetti. A Unified Framework for Analyzing Security of Protocols. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(16), 2001.
10. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. Cryptology ePrint Archive, Report 2006/432, 2006. `http://eprint.iacr.org/`.
11. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. of STOC 2002*, pages 494–503, 2002.
12. N. Chandran, V. Goyal, and A. Sahai. New Constructions for UC Secure Computation Using Tamper-Proof Hardware. In *Proc. of EUROCRYPT 2008*, pages 545–562, 2008.
13. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. of CRYPTO 1994*, pages 174–187. Springer-Verlag, 1994.
14. I. Damgård. In *Advances in Cryptology, Proceedings of the 11th Annual International Cryptology Conference*, CRYPTO 1991, pages 445–456, New York, NY, USA, 1991. Springer-Verlag New York, Inc.
15. I. Damgård, J. Nielsen, and D. Wichs. Isolated proofs of knowledge and isolated zero knowledge. In *Proc. EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 509–526. Springer Berlin Heidelberg, 2008.
16. I. Damgård, J. Nielsen, and D. Wichs. Universally composable multiparty computation with partially isolated parties. In *Proc. of TCC 2009*, volume 5444 of *LNCS*, pages 315–331. Springer Berlin / Heidelberg, 2009.
17. I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 2002, pages 581–596, London, UK, UK, 2002. Springer-Verlag.

18. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proc. of SFCS 1994*, pages 454–465, Washington, DC, USA, 1994. IEEE Computer Society.
19. A. W. Dent. The hardness of the DHK problem in the generic group model, 2006. http://eprint.iacr.org/2006/156.
20. O. Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.
21. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proc. of CSWF 1986*, pages 174–187, oct. 1986.
22. G. P. Hancke. *Security of proximity identification systems*. PhD thesis, July 2009.
23. J. Katz. Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In *Theory and Application of Cryptographic Techniques EUROCRYPT 2007*, pages 115–128, 2007.
24. Y. Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In *Proc. of EUROCRYPT 2011*, pages 446–466, Berlin, Heidelberg, 2011. Springer-Verlag.
25. S. Micali and P. Rogaway. Secure computation (abstract). In *Proc. of CRYPTO 1991*, pages 392–404, London, UK, 1992. Springer-Verlag.
26. J. Monnerat, S. Pasini, and S. Vaudenay. Efficient Deniable Authentication for Signatures, Application to Machine-Readable Travel Document. In *Proceedings of ACNS 2009*, volume 5536 of *LNCS*, pages 272–291. Springer Berlin / Heidelberg, 2009.
27. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO 1991*, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

A Extensions

A.1 A Variant Based on ag-$\mathsf{DDH}_{\mathsf{Gen}}$

We can drop the ag-$\mathsf{DHK0}_{\mathsf{Gen}}$ assumption and solely rely on the ag-$\mathsf{DDH}_{\mathsf{Gen}}$ one. For that, we construct a new Register protocol based on a zero-knowledge proof with the Schnorr Σ-protocol. See Figure 3 on page 15. This would get us closer to [16], where a WI-PoK is used in the key-setup block.

Namely, we enrich the Σ-protocol with a trapdoor commitment Hgen on the challenge c, with the trapdoor σ released at the end. It is a trapdoor in the sense that for all γ and all c', $\mathsf{Equiv}_\sigma(\gamma, c')$ has the same distribution as u and $H_\kappa(c', \mathsf{Equiv}_\sigma(\gamma, c')) = \gamma$. This is quite a standard technique [26]. By making the challenge atomic, we obtain a ZK protocol in a regular sense. It is further straightforward to see that Register satisfies all requirements, based on the ag-$\mathsf{DDH}_{\mathsf{Gen}}$ assumption. To make it authenticating, we can take advantage of the $\mathcal{F}_{\mathsf{atomic}}$ exchange to authenticate X at the same time as the response is sent.

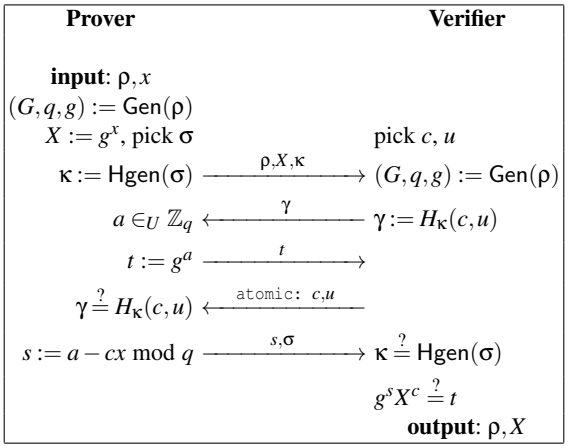

Fig. 3. A ZK Variant for the Register Protocol

In general we prefer to use the ag-$\mathsf{DHK0}_{\mathsf{Gen}}$ to ascertain the private knowledge of sk. This may be more efficient in practice than a full implementation of, e.g., a WI-PoK. It essentially requires the selection of appropriate (and efficient) groups to work in, as done in Example 5.

A.2 Towards $\mathcal{F}_{\mathsf{MCOM}}$

The $\mathcal{F}_{\mathsf{MCOM}}$ functionality is defined as follows:

$\mathsf{Commit}(\mathsf{sid}, R, m)$ **message from** S**.** If sid is not fresh, abort. Otherwise, store $(\mathsf{sid}, S, R, m, \mathsf{sealed})$ and send a $[\mathsf{committed}, \mathsf{sid}, S]$ message to R and to the ideal adversary.

$\mathsf{Open}(\mathsf{sid})$ **message from** S**.** If sid is new or the record $(\mathsf{sid}, S, ., ., .)$ has no matching S, abort. Otherwise, retrieve $(\mathsf{sid}, S, R, m, \mathsf{state})$. If $\mathsf{state} \neq \mathsf{sealed}$, abort. Otherwise, send an $[\mathsf{open}, \mathsf{sid}, m]$ message to R and to the ideal adversary, and replace state by opened in the $(\mathsf{sid}, S, R, m, \mathsf{state})$ entry.

To realize this functionality, we use a similar assumption as in [16]: we assume that a participant plays the role of a trusted certificate authority (who is honest but curious), to whom participants register their keys sk_X and sk_E. The first time a participant is involved in a commitment, he must register his keys to the certificate authority (CA) and get the CA's public key at the same time. The CA would produce a certificate which could be verified with the CA's public key. Then, the Init phase between S and R would reduce to sending and verifying this certificate, without any ideal functionality. Due to the extraction nature of our Register block, all secret keys would become extractable by the ideal adversary and the UC security would still hold.

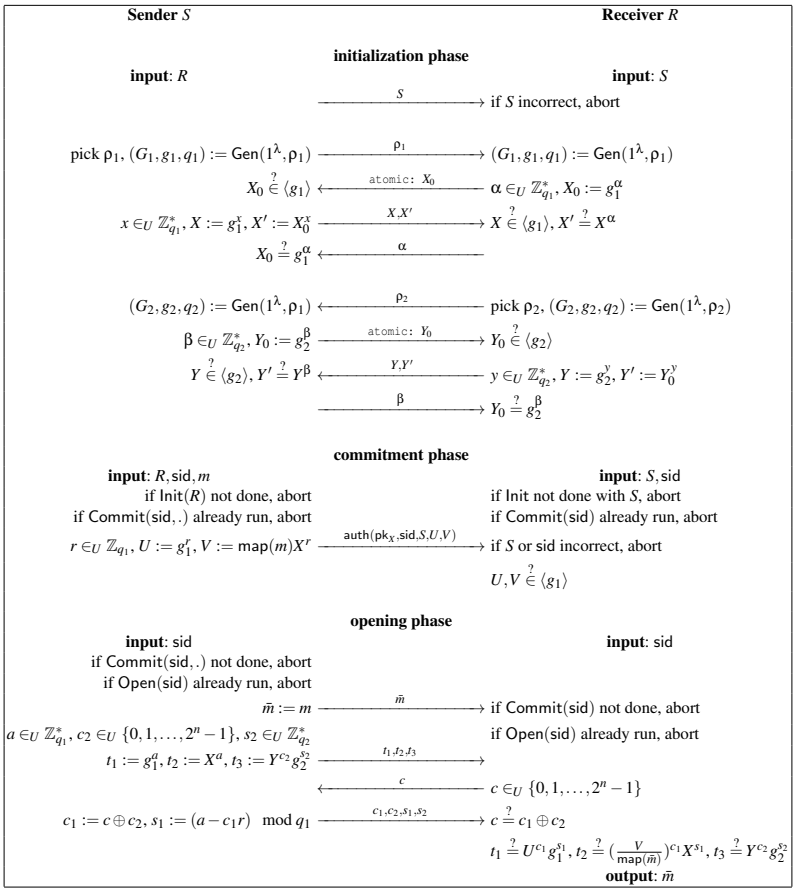

Fig. 4. C-at: A UC-Secure Commitment Protocol with Atomic Exchanges