MLS Group Messaging: How Zero-Knowledge Can Secure Updates

Julien Devigne[1,2], Céline Duguey[1,2(✉)], and Pierre-Alain Fouque[2,3]

[1] DGA Maîtrise de l'information, Bruz, France
julien.devigne@intradef.gouv.fr
[2] Irisa, Rennes, France
{celine.duguey,pierre-alain.fouque}@irisa.fr
[3] Univ Rennes1, CNRS, Rennes, France

Abstract. The Messaging Layer Security (MLS) protocol currently developed by the Internet Engineering Task Force (IETF) aims at providing a secure group messaging solution. MLS aims for end-to-end security, including Forward Secrecy and Post Compromise Secrecy, properties well studied for one-to-one protocols. It proposes a tree-based regular asynchronous update of the group secrets, where a single user can alone perform a complete update. A main drawback is that a malicious user can create a denial of service attack by sending invalid update information.

In this work, we propose a solution to prevent this kind of attacks, giving a checkpoint role to the server that transmits the messages. In our solution, the user sends to the server a proof that the update has been computed correctly, without revealing any information about this update. We use a Zero-Knowledge (ZK) protocol together with verifiable encryption as building blocks. As a main contribution, we provide two different ZK protocols to prove knowledge of the input of a pseudo random function implemented as a circuit, given an algebraic commitment of the output and the input.

Keywords: Cryptographic protocols · Messaging Layer Security - MLS · Secure messaging · Zero-knowledge

1 Introduction

Secure messaging protocols have been widely adopted over the last few years. The privacy offered by encrypted communication seduces billions of users worldwide. A significant number of application providers have settled their security on the Double Ratchet Algorithm [37], often identified as Signal. This protocol provides End-to-End confidentiality, as well as Forward Secrecy (FS) and Post Compromise Secrecy (PCS). The Double Ratchet however is dedicated to one-to-one communications. MLS targets secure *group* messaging and is developed by the IETF. The goal is to obtain similar security properties as those in one-to-one protocols. The group keys are computed and regularly updated in a

E. Bertino et al. (Eds.): ESORICS 2021, LNCS 12973, pp. 587–607, 2021.
https://doi.org/10.1007/978-3-030-88428-4_29

protocol called TreeKEM, based on a tree structure: each member of the group is represented by a tree leaf and the group secret is given by the tree root. To perform an update, a user sends to the other leaves secret information which depend on their position in the tree. In this paper, we are concerned with an open problem identified in the MLS draft: how to be sure that each user receives a valid update information? In other words: how to be sure that the updating user is not cheating? The current protocol provides verification elements for each user to check whether the update he received is valid, but it does not prevent a malicious updater to send misformed update information to all or part of the group. Such attacks would prevent the updating process, seriously damaging FS and PCS properties, that seduce the users. Consequently, we propose a solution in which the users only receive valid updates: the server which transmits the update messages can check their validity before forwarding them. The server is given a check-point role and has no more power to create an update, malicious or not, than in the original protocol. Hence we only add a layer of security, through the server, without modifying the core of MLS. A main building block of our solution is a ZK protocol, inspired from the recent multi party computation (MPC) in the head solutions [27,30,34].

Our Contribution. As a first contribution, we show how to combine a ZK protocol with a verifiable encryption solution to solve the open problem identified in the IETF draft for MLS, in a light version of the protocol. The idea is to enable an intermediate server to perform a blind verification (on encrypted and committed data) that each update information sent by the updater is correctly computed. The ZK proof is provided on a statement that mixes a circuit evaluation (an HKDF derivation, defined in [36]) and an algebraic commitment (typically a Pedersen commitment, described in [38]). The verifiable encryption scheme proves to the server (*i.e.* the verifier) that the encrypted data is the one that is committed to and verified in the proof.

As a second contribution, we propose two ZK protocols for statements that compose an algebraic commitment and the circuit computation of a pseudo-random function family (PRF) f. More precisely, we prove the knowledge of a secret witness x such that public commitments C_x, C_y are algebraic commitments of x and $f(x)$, with $f(x)$ remaining secret. Our approach is based on the MPC in the head paradigm, introduced in [30]. We consider the recent ZK proof system ZKBoo [27] and its improvements ZKBoo++ [18] as well as KKW [34]. Our first approach is directly inspired by a recent work of Backes *et al.* [6]. The first step is to provide commitments to the bits of the secret input and output. Then, we call the algebraic properties of the commitment scheme to link those bits with values used in the circuit proof. Our second approach consists in considering a Boolean circuit that computes a tag $t = f(x) + ax$. The proof on this circuit binds the secret input and output to the tag value. In a second step, we use the homomorphic properties of the commitment to show that the committed values are also bound by t. Finally, we invoke properties of the function f to show that their exists only one solution to the tag equation.

Previous works, in particular [19] and [1], have proposed solutions for such algebraic and circuit composition statements. However, the former is inherently interactive and the latter uses SNARKs, where the burden of the proof is mainly on the prover side, which does not fit our use case as smartphones have resources to be saved.

Related Work. *Secure Messaging,* and more particularly Ratcheted Key Exchange (RKE), have been widely studied (*e.g.* [3,9,31,33,39]) since the first analysis of the Signal protocol [22]. Literature for the group version is more limited. In [21] Cohn Gordon *et al.* introduced the notion of Asynchronous Ratcheted Trees (ART). These ART are Diffie Hellman based binary trees in which the update process of a node involves entropy coming from both its children. In MLS, the underlying TreeKEM protocol is inspired by ART. A main difference is that a single leaf can generate the update data for each of its ancestor nodes. TreeKEM has been initially formalized in the technical paper [11] and has then evolved to reach the actual description available on the prevailing draft 11 [7]. Alwen *et al.* formalize in [4] a Continuous Group Key Agreement (CGKA) derived from the two-party Continuous Key Agreement defined in [3]. They provide a security model for CGKA and show that TreeKem does not achieve optimal FS and PCS security, but prove that using updatable public key encryption can lead to a better security. Our solution is compatible with this improvement. In [2], Alwen *et al.* focus on the addition and revocation process. Finally, Brzuska *et al.* provide in [12] an analyse of the current draft 11, considering both TreeKem and the Key Schedule on top of it.

Zero-Knowledge (ZK) proofs, have proved to be a powerful tool in cryptography, since their conception in the mid 1980s. It has been shown that ZK proofs exist for any NP language. However, efficient ZK protocols are designed for a small class of language and do not extend to any NP languages. Sigma protocols (Σ-protocols), clearly described in [23], are very efficient for proving algebraic relations, whereas other protocols have been designed for proving statements that can be expressed as a circuit. Among them, Garbled Circuits based schemes, as introduced in [32], which are inherently interactive, and SNARKs, as designed in [26,28,35]. SNARKs are non-interactive arguments (with computational soundness) of knowledge with small proofs and light verification: the burden of the proof is on the prover side. They are proven secure in the common reference string (CRS) model: a common trusted public input has to be shared by the prover and the verifier. Practical implementations based on pairings are in use in real life protocols such as cryptocurrencies. STARK proofs [10] remove the CRS requirement but the prover's algebraic computations are still linear in the circuit size. The MPC in the head paradigm, introduced in [30], leads to very efficient proof system without CRS. The seminal paper [27] introducing ZKBoo proposes the first efficient ZK proof of a hash function computation. Further works significantly optimize the efficiency, such as Ligero [5], ZKBoo++ [18] and KKW [34] (developed for the post quantum signature scheme Picnic) or [29].

In real life however, many applications need to provide proofs on statements that mix algebraic and non algebraic parts. Expressing the algebraic part as

a circuit would considerably increase the circuit size and reduce the efficiency. One could express each gate of a circuit as an algebraic relation that can be proven with a Σ-protocol, but this solution is clearly non desirable as circuits for hashing may have thousands of gates. Considering this, combining efficiently algebraic and non algebraic proofs has revealed to be an important challenge.

In [19], Chase *et al.* propose two constructions, based on Garbled Circuits, to provide a circuit proof on a committed input. Our solutions are close to theirs in the sense that their first proposal uses bit wise commitment on an secret input, and their second proposal includes a one-time mac computation in the circuit to be garbled. However, their proposal heavily relies on the garbling protocol and can not be transposed to the non interactive setting.

More recently, Agrawal *et al.* in [1] propose a solution for modular composition of algebraic and non algebraic proofs. Their solution is non interactive, based on Sigma protocols and QAP-based SNARKs. As explained in their work, the "key ingredient [they] need from a SNARK construction is that the proof contains a multi-exponentiation of the input/output". They compose it with a proof that the exponents in a multi-exponentiation correspond to values committed to in a collection of commitments. From this result, they show how to obtain proofs for AND, OR and composition of two statements, either algebraic or circuit. The small proofs and the light verification step of SNARKS are desirable for privacy-preserving credentials or crypto-currencies proofs of solvency. But the prover's high computational effort is not adapted to our application where the verifier turns out to have a larger computation power than the prover.

Finally, Backes *et al.* propose in [6] an extended version of ZKBoo++ that allows algebraic commitments on the secret input of the circuit. Their protocol is non interactive and the computational cost is balanced between the prover and the verifier. Their solution requires to commit to each bit of the secret input and to commit to internal values of the ZKBoo++ circuit proof. We extend their result to the case of a committed output in our first zero-knowledge solution. We focus on the MPC in the head paradigm in order to provide proofs in which the amount of work is balanced between both parties.

Organization of the Paper. In Sect. 2, we recall the definitions concerning ZK proofs, commitment schemes, and verifiable encryption. In Sect. 3, we present the protocol MLS, focusing on the process to update the group secret, and we describe our solution to improve the security of the update mechanism. The Sect. 4 is dedicated to our two protocols, `CopraZK` and (bitwise) `ComInOutZK`, for proving knowledge of the preimage of a PRF function when only commitments of the input and the output are publicly available.

2 Backgrounds

Zero-Knowledge. Consider an NP relation $\mathcal{R}$, *i.e.* given a witness w and an input x, $\mathcal{R}(x, w) = 1$ can be decided in polynomial time. Let $\mathcal{L}$ be the language associated to $\mathcal{R}$, $\mathcal{L} = \{x | \exists w \text{ such that } \mathcal{R}(x, w) = 1\}$. A ZK proof of knowledge

for $\mathcal{L}$ allows a prover to convince a verifier, that he knows a witness w for x, without revealing w. It shall be correct (if the prover and the verifier are honest, the verifier always accepts), sound (an efficient extractor that interacts with a corrupted prover can exhibit a valid witness except with negligible probability), and zero-knowledge (no information on w leaks from the proof). Following the notation of [14], we write $\mathsf{PK}\{w_1, \dots w_s : \mathcal{R}(w_1, \dots, w_s, x_1 \dots x_t) = 1\}$ to denote the proof of knowledge of the secret witnesses $w_1, \dots w_s$ that satisfy the relation $\mathcal{R}$ with the public values $x_1, \dots, x_t$.

Σ-Protocols. These are specific three moves proofs of knowledge. The prover first sends a commitment value q, receives a challenge e and answers with a value t. Given (x, a, e, t), the verifier accepts or not. Those protocol provide correctness, s-special soundness (given s transcripts with common commitments and distinct challenges, one can extract a witness) and honest-verifier ZK (the verifier is suppose to generate the challenge honestly). They can be turned into a non-interactive proof using Fiat-Shamir transformation [25]. In this case the special-honest verifier clause disappears and the verification step consists in reconstructing the challenge from the received data.

Commitment Schemes. A commitment scheme involves a *Committer* and a *Receiver* who share public parameters. On entrance a message x and an additional opening information r, the commitment protocol produces a value $c = \mathsf{Com}(x, r)$ such that c shall not reveal any information about x; this is the hiding property. The *Committer* can open its commitment c by revealing r and x with the property that only the secret x shall produce a valid opening for c; this is the binding property. For our first ZK protocol, we will require an extra property, called equivocality. A commitment is equivocable if there exists a trapdoor T such that, given a commitment C, its opening information, and T, it is possible to open C to any value. Equivocality comes with a specific extractor that, given two different openings (x_1, r_1), (x_2, r_2) to a single commitment C, can extract the trapdoor T. The Pedersen commitment scheme [38] is an equivocable scheme with unconditional hiding and computational binding. It is routinely used because it interacts nicely with linear relations. This scheme is defined as follows: let $\mathbb{G}$ be a cyclic group of prime order q, P a generator and $Q \in \mathbb{G}$ such that the discrete log of Q in base P is unknown. Then, $\mathsf{Com}(x) = xP + rQ$ where r is sampled at random in $\mathbb{Z}_q$. Let C_1, C_2 be commitments to values x_1, x_2. If $a, b \in \mathbb{Z}_q$ are public values, then one can efficiently prove the following: $PK\{x_1, x_2, r_1, r_2 : C_1 = x_1P + r_1Q \wedge C_2 = x_2P + r_2Q \wedge ax_1 + x_2 = b \mod q\}$. The trapdoor for equivocality is given by the discrete log of Q in base P.

MPC in the Head. Ishai *et al.* introduced in [30] a new paradigm for ZK proofs, called MPC in the head. This solution reveals to be very competitive in terms of efficiency for ZK proofs performed on circuits. The idea is that the prover performs a virtual MPC and obtains several views. He commits to these views and opens only a sub-part of them required by the verifier. ZKBoo [27] generalizes IKOS to any relation $\mathcal{R}_\phi$ defined by a function $\phi : X \to Y$ ($\mathcal{R}_\phi(y, x) \leftrightarrow y = \phi(x)$), as long as the function ϕ can be computed in a specific

manner identified as a (2,3)-decomposition. The prover first shares its secret input x into $(x_1, x_2, x_3) = \mathsf{Share}(x)$ such that $x = x_1 \oplus x_2 \oplus x_3$. Then he runs the MPC and obtains three distinct views w_1, w_2, w_3 and from each of this view he gets an output share $y_i = \mathsf{Output}(w_i), i \in \{1, 2, 3\}$ such that $y_1 \oplus y_2 \oplus y_3 = \phi(x)$.

Verifiable Encryption. Verifiable encryption aims at convincing a verifier that an encrypted data satisfies some properties without leaking any information about the data itself. In such 2-party protocol, a prover and a verifier share in a common input string a public key encryption scheme Enc, a public key pk for Enc, and a public value y. At the end, the verifier either accepts and obtains the encryption of a secret value x under pk such that x and y verify some relation $\mathcal{R}$ or rejects. It is worth noticing that the prover does not need the secret key sk, that usually belongs to a third party. Verifiable encryption often appears in the domain of anonymous credentials, fair exchange signatures, or verifiable secret sharing [40]. In [13], Camenish and Damgård describe how to provide a proof that an encrypted value is a valid signature, using any semantically secure encryption scheme. The idea is to take advantage of the Σ-protocol for a relation $\mathcal{R}(x, y)$, to prove that an encrypted value is the witness x for this relation. In our application, we need to prove that the data that is encrypted, x, is the one that is linked by a Pedersen commitment $C_x = y$. As a Pedersen commitment comes with an associated Σ-protocol, the Camenish-Damgård scheme applies naturally. There are interesting ways towards more efficient schemes, *e.g.* [20] or [16]. However, the main benefit of the Camenish-Damgård solution is that we can still use the encryption scheme required in MLS specification. We denote by $\mathsf{VerifEnc}_{\mathsf{Enc},pk}(m : r)$ the encryption of a message m (using randomness r) under the public key pk with the encryption scheme Enc and the associated proof. We omit the randomness r when it is not necessary to explicitly mention it.

3 MLS Updates

We explain how the MLS update mechanism works and our more secure solution.

3.1 Message Layer Security

MLS is a protocol currently under development by the IETF to provide an End-to-End secure group messaging application. The idea is to enable a group of users to share a common secret that can be updated regularly by any member. One of the open issues in the IETF draft is that the validity of an update message can only be checked *after* it has been received. This open issue is clearly identified in the current draft 11 ([7], Sect. 15.5). However, to our knowledge, there is still no solution to this problem. Currently in MLS, the authors require an hybrid public key encryption (HPKE) scheme, as designed in [8], composed of a key encapsulation mechanism (KEM) to transmit a symmetric key k and an AEAD encryption scheme that encrypts the data under k, as well as a key derivation function. In the rest of this work, we denote by $\mathsf{Enc}_{pk}(m : r)$ the HPKE encryption of a message m under the public key pk using randomness r.

The asymmetric part of Enc is based on an elliptic curve E defined on a finite field $\mathbb{Z}/p\mathbb{Z}$ with base point P of order a prime q. MLS also supposes the existence of a broadcast channel for each group, which distributes the messages to each group member, conserving the order.

TreeKem. MLS key exchange TreeKem is based on a binary tree structure (Fig. 1) where users correspond to leaves and each node is associated to a secret value. Each user U has a long term identity signing key and an initial key package for the encryption scheme Enc (both certified by a PKI). We will simply represent the key package as a public/private key pair (pk_U, sk_U).

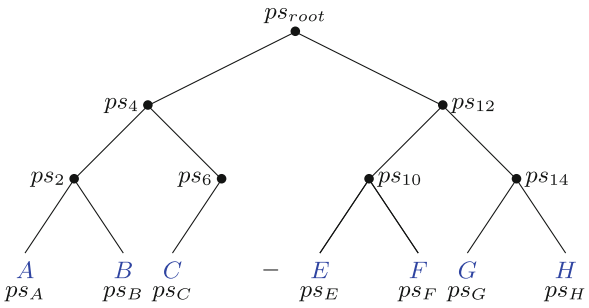

Fig. 1. A view of the MLS tree. Nodes are implicitly numbered from left to right, independently from their height. Leaves are associated to a user represented as a letter. Each node i has a secret ps_i. A leaf secret is indexed with its user name.

The group key is derived from the root secret. Each child node knows the secret of each of its ancestors and only of its ancestors. To each node i corresponds a path secret ps_i and a secret and public key $sk_i, pk_i = \mathsf{deriveKeyPair}(ps_i)$ (in the original protocol the keys are derived from an intermediate node secret ns_i itself derived from ps_i. We present a lighter version for the simplicity of the exposure but the complete version is compatible with our solution.)

The derivation depends on the elliptic curve (see Appendix A). We define, w.l.o.g, $(sk_i, pk_i) = \mathsf{deriveKeyPair}(ps_i) = (\mathsf{deriveSK}(ps_i), \mathsf{deriveSK}(ps_i)P)$ where deriveSK is a PRF. A user knows the secrets ps_i and sk_i in his direct path, composed of himself and his direct ancestors. Moreover, each user keeps an up-to-date global view of the tree, as a hash value of each node's public information.

Updates. To update the tree, a user B generates a new secret ps'_B. The path secrets in the direct path will be successively derived from ps'_B. We note $\mathsf{H}_p(ps_i)$ for the function $\mathsf{HKDF} - \mathsf{expand}(ps_i, \text{“}path\text{”}, \text{“”}, Hash.length)$. The update mechanism is given in Fig. 2. When B updates its secret $ps_B \rightarrow ps'_B$, he first computes the new node data for each node on his path:

- $ps'_2 = \mathsf{H}_p(ps'_B)$, $pk_2 = \mathsf{deriveSK}(ps'_2) \cdot P$
- $ps'_4 = \mathsf{H}_p(ps'_2)$, $pk_4 = \mathsf{deriveSK}(ps'4) \cdot P$
- $ps'_{root} = \mathsf{H}_p(ps'_4)$, $pk_{root} = \mathsf{deriveSK}(ps'_{root}) \cdot P$

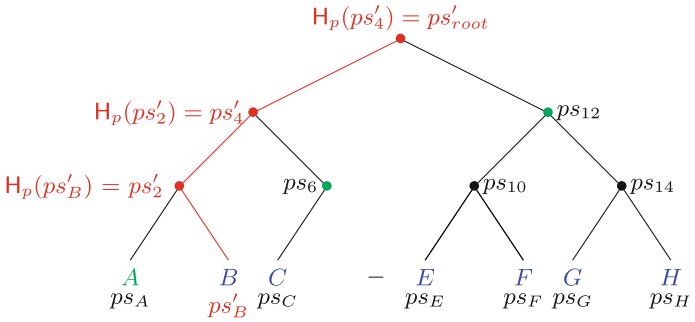

Fig. 2. Update process. User B updates its secrets. Path secrets are updated along its direct path (in red). The update secrets are sent to its copath nodes (in green). (Color figure online)

Then he sends for each node on its copath the necessary secret material for the users under this node to perform the same update. Following our example in Fig. 2, B has to send ps'_2 to A, ps'_4 to nodes C and 6 and ps'_{root} to nodes $E, 10, F, 12, G, 14, H$. As a child knows the secret key sk_i for each of its ancestors, B will only have to encrypt ps'_2 under pk_A, ps'_4 under pk_6 and ps'_{root} under pk_{12}.

From ps'_2 (respectively ps'_4), A (resp. C) shall be able to compute the root secret. From this root secret is derived an epoch secret S_{E+1}. Before sending his Commit, B computes S_{E+1} and uses it to produce a confirmation key. This value shall enable A and C to check that they have derived the correct root secret and so, that they received a correct update. Other mechanisms such as the transmission of the updated view of the tree, or of intermediate hash values are provided for a user to check that he received a correct update. All those mechanisms enable a verification after receiving the update information.

Hence, either the update is adopted only once some members have confirmed that they received a valid update - at least one member in the subtree of each node in the update copath, as recommended in draft 11. This can imply a huge latency, if some users are seldom online, and non valid updates can lead to a denial of service. Or the update is validated without feedback and the users that received non valid secret values are ejected from the group *de facto*. In both case, this seriously hampers with the security of the service provided by the protocol.

3.2 Securing MLS Updates

We now explain how to combine a ZK protocol and a verifiable encryption to secure the update process in MLS. We first focus on a single step of the update process (a user updates his direct parent) and then explain how this solution can be extended to the global tree.

Server-Checking in MLS. As described in Fig. 2, let assume that B generates a new secret ps'_B and computes:

– $\mathsf{deriveKeyPair}(ps'_B)$ to obtain a new key package and $C_B = \mathsf{Com}(ps_{B'}, r_{B'})$;
– $ps'_2 = \mathsf{H}_p(ps'_B)$ the new secret for node 2 and $C_2 = \mathsf{Com}(\mathsf{H}_p(ps'_B), r_2)$;
– $(sk'_2, pk'_2) = \mathsf{deriveKeyPair}(ps'_2)$ the new keys for node 2 and the corresponding $C_{sk'_2} = \mathsf{Com}(\mathsf{deriveSK}(\mathsf{H}_p(ps'_B)), r_{sk'_2})$.

Suppose there exists a ZK protocol which, given public values C_x and C_y, provides the following proof: $\mathsf{PK}\{x, r_x, r_y : C_x = \mathsf{Com}(x, r_x) \wedge C_y = \mathsf{Com}(f(x), r_y)\}$ for any PRF f. Then B can send to the server the public values C_B, C_2, $C_{sk'_2}$, pk'_2 together with a proof $\Pi_2 = \mathsf{PK}\{ps'_B, r_{B'}, r_2, r_{sk'_2} : C_B = \mathsf{Com}(ps'_B, r_{B'}) \wedge C_2 = \mathsf{Com}(\mathsf{H}_p(ps'_B), r_2) \wedge C_{sk'_2} = \mathsf{Com}(\mathsf{deriveSK}(\mathsf{H}_p(ps'_B)), r_{sk'_2}) \wedge pk'_2 = \mathsf{deriveSK}(ps'_2)P\}$ (the last part of the proof being a classic discrete log proof).

On another side, verifiable encryption (detailed in Sect. 2) allows to link the message encrypted with $\mathsf{VerifEnc}$ with the data committed in C_2. To sum up, B will send for a node update, the public values C_B, C_2, $C_{sk'_2}$, and pk'_2, the proof Π_2 together with $\mathsf{VerifEnc}_{\mathsf{Enc},pk_A}(ps'_2)$. If the server accepts the proof, then he transmits the public key pk'_2 as well as $\mathsf{VerifEnc}_{\mathsf{Enc},pk_A}(ps'_2)$ to A.

To extend the proof to the complete tree, one has to repeat the above steps for each level. To certify the update value ps'_4 corresponding to the parent node 4, B will send the server values C_4, $C_{sk'_4}$, pk'_4, the proof $\Pi_4 = PK\{ps'_2, r_2, r_4, r_{sk'_4}: C_2 = \mathsf{Com}((ps'_2), r_2) \wedge C_4 = \mathsf{Com}(\mathsf{H}_p(ps'_2), r_4) \wedge C_{sk_4} = \mathsf{Com}(\mathsf{deriveSK}(\mathsf{H}_p(ps'_2)), r_{sk'_4}) \wedge pk'_4 = \mathsf{deriveSK}(\mathsf{H}_p(ps'_2))P\}$ together with $\mathsf{VerifEnc}_{\mathsf{Enc},pk_6}(ps'_4)$. The crucial point is that, as the commitment C_2 is linked with ps'_B in Π_2, it can be used as a base value for Π_4 and so on. Some special care must be taken as we commit, in a group of order q prime, to an element $sk \in \{0,1\}^{256}$ that does not lie naturally in $\mathbb{Z}/q\mathbb{Z}$. We explain how to handle with this in Appendix A.

About the Server. Several reasons appear for calling on a third party. Firstly, this central node with the largest computational power is the one that can discard invalid updates with the most efficiency. If one relies on users to check for the validity of the data they received, this means that one must wait for each user to process the update and to send back an acknowledgement. As a user can be off-line for a long time, this can be very inefficient. Another solution would be to allow users to adopt the update as soon as they are individually convinced it is correct, while providing a "backup solution". This would probably imply keeping old keys and drastically impoverish FS.

Secondly, in MLS architecture, all the update encrypted messages are gathered and sent as one big message to all the users. It may be of interest to think of a solution where only the needed encryption is sent to a specific user. In this case, only the server will see all the messages together. He is then the only one able to perform a verification on a global proof to see whether all the updates are correctly generated from a single secret seed.

4 ZK for a PRF on Committed Input and Output

In this section, we provide two protocols to prove the knowledge of an input x and randoms r_x, r_y, such that, for a public values C_x, C_y, and a function f

evaluated as a circuit, $C_x = \mathsf{Com}(x, r_x)$ and $C_y = \mathsf{Com}(f(x), r_y)$. Recall that efficient ZK proofs for a function evaluation are operated on a circuit, whereas efficient commitments are algebraic. Consequently, we want to achieve the best of both worlds by combining a proof on a circuit and algebraic commitments.

Our first solution, `ComInOutZK` (Committed Input and Output ZK) is directly inspired from [6], which provides a proof of a circuit evaluation on a committed input and public output. We extend their work to a committed output.

Our second solution, `CopraZK` (Commitment and PRF alternative ZK), requires some specific properties on the function f. We consider the circuit that computes $f(x, m)+\alpha x$ where x is the key of the PRF and α is determined by the commitment values. Calling some PRF-related properties, and the homomorphic properties of the commitment, we show that the values committed in C_x and C_y must be those that appear in the circuit evaluation.

We compare in Table 1 our two solutions with the SNARK based solution of [1]. `CopraZK` adds a negligible number of algebraic operations. The prover performs 4 multiplications on the curve (public key operations) and 8 computations in $\mathbb{Z}/q\mathbb{Z}$ (symmetric operations). For the verifier, 6 computations on the curve and 2 in $\mathbb{Z}/q\mathbb{Z}$ are needed. Considering ZKBoo, the prover effort is $\mathcal{O}(\sigma|F|)$ symmetric operations, where $|F|$ is the number of AND gates of the circuit and σ the number of rounds. Our solution requires $\mathcal{O}(\sigma(|F| + |mod|)) + 8$ symmetric operations and 4 public key operations, where $|mod|$ is the size of the circuit for a modular addition which is negligible compared to $|F|$. The computational cost is dominated by the symmetric part and finally, our solution requires on the prover side ($\mathcal{O}(\sigma|F|)$ symmetric operations. The size of the proof and the work on the verifier's side are also dominated by the circuit part. One inconvenient is that the security proof requires non usual hypothesis on the function f.

On the opposite side, `ComInOutZK` is valid for any circuit, only requires equivocality of the commitment scheme, which is a common hypothesis, and leaves the circuit evaluation untouched. But it requires a non negligible number of algebraic commitments. Considering $|x|$ (respectively $|y|$) the bit size of the input (of the output), we obtain on the prover side $\mathcal{O}(|x| + |y| + 2\sigma)$ public key operations and $\mathcal{O}(\sigma|F|)$ symmetric operations. The verifier's work is equivalent. The proof size of ZKBoo is augmented with $\mathcal{O}(|x| + |y| + 6\sigma)$ curve points which is asymptotically $\mathcal{O}((|x| + |y| + \lambda)\lambda)$ as σ augments with λ.

On the Challenge Size. When we expose our solutions, in both case we mention a unique challenge, that is used for the algebraic Σ-protocol and for the ZKBoo proof. This means that the challenge space size for the Σ-protocol is 3 and that we shall perform $\lambda/3$ rounds to obtain a soundness error in $2^{-\lambda}$. The Σ-protocol can benefit from a larger challenge space, that allows for a single round. As explained in [6], it is possible to define distinct challenges $e_\rho \in \{1, 2, 3\}$ for each ZKBoo round and a global challenge $e = \sum_{i=1}^{\sigma} 3^i e_i$ for the algebraic Σ-protocols, hence the algebraic part of the proof can be performed a single time.

Table 1. Efficiency of the different solutions for circuit proof on committed input and output. *pub* stands for the cost of a public key operation (multiplication on the curve), while *sym* stands for the cost of a symmetric operation. $|F|$ is the circuit size, $|x|$ the input size and $|y|$ the output size. In most applications, $|F| \gg (|x|, |y|, \lambda)$.

	Non interactive	No CRS	Prover's work	Verifier's work	Proof size
SNARK based [1]	Yes	No	$\mathcal{O}((\lvert F\rvert + \lambda) \cdot pub)$	$\mathcal{O}((\lvert x\rvert + \lvert y\rvert + \lambda) \cdot pub)$	λ
`CopraZK`	Yes	Yes	$\mathcal{O}(\lvert F\rvert\lambda \cdot sym)$	$\mathcal{O}(\lvert F\rvert\lambda \cdot sym)$	$\mathcal{O}(\lvert F\rvert\lambda)$
`ComInOutZK`	Yes	Yes	$\mathcal{O}(\lvert F\rvert\lambda \cdot sym + (\lvert x\rvert + \lvert y\rvert + \lambda) \cdot pub)$	$\mathcal{O}(\lvert F\rvert\lambda \cdot sym + (\lvert x\rvert + \lvert y\rvert + \lambda) \cdot pub)$	$\mathcal{O}((\lvert F\rvert + \lvert x\rvert + \lvert y\rvert + \lambda)\lambda)$

4.1 `ComInOutZK`: A Bit-Wise Solution

In [6], the authors propose a non interactive proof $PK\{x : C_x = Com(x, r_x) \wedge y = f(x)\}$ based on bit commitments and ZKBoo++. Their optimized solution increases the ZKBoo++ prover's and verifier's work with $\mathcal{O}(|x| + \sigma)$ exponentiations and multiplications on the group $\mathbb{G}$ of order q chosen for the commitment, where $|x|$ is the number of bits of the input x and σ is the number of rounds in ZKBoo++. The proof size grows by $\mathcal{O}(|x| + \sigma)$ group elements and $\mathcal{O}(|x| + \sigma)$ elements in $\mathbb{Z}/q\mathbb{Z}$. We adapt this strategy in the case of a committed output. As the output of the circuit, y, shall remain secret, we will not be able to call ZKBoo++ as a full black box. This is of prime importance when we prove the zero-knowledge property.

The work of Backes *et al.* and our extension rely on a result given by the homomorphic property of a commitment scheme such as Pedersen scheme. For any scalar k, and any two commitments $\mathsf{Com}(x, r_x)$, $\mathsf{Com}(y, r_y)$, $k \cdot \mathsf{Com}(x, r_x) + \mathsf{Com}(y, r_y) = \mathsf{Com}(kx + y, kr_x + r_y)$. For any commitment $C_b = \mathsf{Com}(b, r_b)$ to a secret bit b and any public bit β, one can easily compute the commitment of $b \oplus \beta$ as follows: if $\beta = 0$, $C_{b\oplus\beta} = C_b$ and if $\beta = 1$ then $C_{b\oplus\beta} = \mathsf{Com}(1, 0) - C_b = \mathsf{Com}(1 - b, -r_b)$. For any $x = \sum_{i=0}^{|x|-1} 2^i x[i]$, denote $C_{x[i]} = Com(x[i], r_{x[i]})$ a commitment to the i-th bit of x. Then $\sum_{i=0}^{|x|-1} 2^i C_{x[i]}$ is a valid commitment to x with opening randomness $\sum_{i=0}^{|x|-1} 2^i r_{x[i]}$. And one can easily compute a commitment to $x \oplus \beta$ for an element β as $C_{x\oplus\beta} = \sum_{i=0}^{|x|-1} 2^i C_{x[i]\oplus\beta[i]}$, with opening randomness $\sum_{i=0}^{|x|-1} 2^i (-1)^{\beta[i]} r_{x[i]}$.

We describe in Fig. 3 the protocol on committed output only (`ComOutZK`), for readability reasons. Combining Backes *et al.* protocol for committed input and ours for committed output leads to `ComInOutZK`.

Let f be a function: $\mathbb{Z}_2^\ell \to \mathbb{Z}_2^\ell$, $\mathbb{G}$ be a group of prime order q, such that $2^\ell \leq p$. There is a natural embedding $\mathbb{Z}_2^\ell \hookrightarrow \mathbb{G}$. Let P be a generator for this

The *Prover* knows x, $y = f(x)$, and r_y such that $C_y = \mathsf{Com}(f(x), r_y)$. The *Verifier* knows the statement C_y.

Prover

Commit phase

1. samples random $r_{y[j]}$ and commits to the bits of y : $C_{y[j]} = \mathsf{Com}(y[j], r_{y[j]})$ for $j \in [0, |y|]$.
2. computes the commit phase a_{Π_j} for the proofs $\Pi_j = PK\{y[j], r_{y[j]} : C_{y[j]} = \mathsf{Com}(y[j], r_{y[j]}) \wedge y_[j] \in \{0, 1\}\}$ for $j \in [0, |y|]$.

for $\rho \in [1, \sigma]$:

3. samples random seeds k_1^ρ, k_2^ρ, k_3^ρ.
4. generates the shares $x_1^\rho, x_2^\rho, x_3^\rho = \mathsf{Share}(x, k_1^\rho, k_2^\rho)$ such that $x = x_1^\rho \oplus x_2^\rho \oplus x_3^\rho$.
5. simulates the MPC to obtain three views $w_1^\rho, w_2^\rho, w_3^\rho$.
6. evaluates $y_i^\rho = \mathsf{Output}(w_i^\rho)$, $i \in \{1, 2, 3\}$.
7. commits to the views : $c_i^\rho = \mathsf{h}(w_i^\rho, k_i^\rho)$, $i \in \{1, 2, 3\}$.
8. samples random $r_{y_i^\rho}$ and commits to the outputs : $C_{y_i^\rho} = \mathsf{Com}(y_i^\rho, r_{y_i^\rho})$, $i \in \{1, 2, 3\}$.

$a = ((C_{y_1^\rho}, C_{y_2^\rho}, C_{y_3^\rho}, c_1^\rho, c_2^\rho, c_3^\rho)_\sigma, (C_{y[j]})_{|y|}, (a_{\Pi_j})_{|y|})$

Challenge : $e = \mathsf{h}(a)$

Response phase

1. computes the responses z_{Π_j} for the proofs Π_j

for $\rho \in [1, \sigma]$:

2. $b^\rho = (C_{y_{e+2}^\rho}, c_{e+2}^\rho)$
3. $z^\rho = (w_{e+1}^\rho, k_e^\rho, k_{e+1}^\rho, r_{y_e^\rho}, r_{y_{e+1}^\rho})$
4. $\beta^\rho = y_e^\rho \oplus y_{e+1}^\rho$
5. $C_z^\rho = \sum_{i=0}^{|y|-1} 2^i C_{y[i] \oplus \beta^\rho[i]}$
6. $r_z^\rho = r_{y_{e+2}^\rho} - \sum_{i=0}^{|y|-1} 2^i (-1)^{\beta[i]} r_{y[i]}$

return $p = (e, (b^\rho, z^\rho, r_z^\rho)_\rho), (z_{\Pi_j})_j$

. . . . . .

Verifier(a, p)

1. Parses p as e, $(b^\rho, z^\rho, r_z^\rho)_\sigma$
2. Parses a as $(C_{y_1^\rho}, C_{y_2^\rho}, C_{y_3^\rho}, c_1^\rho, c_2^\rho, c_3^\rho)_\sigma, (C_{y[j]})_|y|, a_{\Pi_j})$
3. Reconstructs the proof Π_j (computes a_Π from $(z_{\Pi_j})_j$)
4. Rejects if $C_y \neq \sum_{i=0}^{|y|-1} 2^i C_{y[i]}$

for $\rho \in [1, \sigma]$:

5. runs the MPC protocol to reconstruct w_e^ρ from $w_{e+1}^\rho, k_e^\rho, k_{e+1}^\rho$
6. obtains $y_e^\rho = \mathsf{Output}(w_e)$, $y_{e+1}^\rho = \mathsf{Output}(w_{e+1})$
7. Computes $\beta^\rho = y_e \oplus y_{e+1}$
8. Computes $C_z^\rho = \sum_{i=0}^{|y|-1} 2^i C_{y[i] \oplus \beta^\rho[i]}$
9. Rejects if $C_{y_{e+2}^\rho} \neq \mathsf{Com}(0, r_z) + C_z^\rho$

Reconstructs a and reject if $e \neq h(a)$

Fig. 3. The `ComOutZK` protocol. Combining this protocol with the same mechanism on the input given in [6] leads to `ComInOutZK`. The reconstruct step of the verification consists in computing the commitment a from the response data and check its validity with the challenge. Only the values in a that can not be reconstructed need being sent.

group and Q an element of $\mathbb{G}$ such that $\log_P(Q)$ is unknown. We consider a hash function $\mathsf{h} : \mathbb{Z}_2^* \to \mathbb{Z}_2^{\ell *}$ and Com be the Pedersen commitment scheme.

The Theorem 1 states the security of our bitwise solution.

Theorem 1. *Given that ZKBoo and the Π_j are Σ protocols with 3-special soundness and honest verifier Zero Knowledge property, and* Com *is a homomorphic and equivocal commitment scheme, then the protocol described in Fig. 3 is a Σ-protocol with 3-special soundness and honest verifier property.*

Proof of the Security of `ComInOutZK`. We study separately the three properties a Σ-protocol should verify.

Correctness. Assuming the *Prover* and the *Verifier* execute the protocol as described, the *Verifier* never meets a rejection cause and then always accepts.

Soundness. Consider an algorithm $\mathcal{E}xt$ that has access to three distinct accepting executions of the protocol on the same commit phase: (a, e_1, p_1), (a, e_2, p_2), and (a, e_3, p_3), $e_1 \neq e_2 \neq e_3$, for a public statement C_y. We show that $\mathcal{E}xt$ can exhibit a witness (x^*, r^*) such that $C_y = \mathsf{Com}(f(x^*), r^*)$. We can not directly call the Extractor from ZKBoo++ as we do not exactly execute ZKBoo. In our protocol, the *Verifier* does not have access to the output of the circuit. However, we show that this difference does not prevent $\mathcal{E}xt$ from succeeding. We describe in the following how $\mathcal{E}xt$ works.

Firstly, from the distinct transcripts $\mathcal{E}xt$ can obtain three pairs of shares x_{e_1}, x_{e_1+1}, x_{e_2}, x_{e_2+1}, x_{e_3}, x_{e_3+1}. He also gets three pairs of output values y_{e_1}, y_{e_1+1}, y_{e_2}, y_{e_2+1}, y_{e_3}, y_{e_3+1} and the corresponding randomness $r_{y_{e_1}}$, $r_{y_{e_1+1}}$, $r_{y_{e_2}}$, $r_{y_{e_2+1}}$, $r_{y_{e_3}}$, $r_{y_{e_3+1}}$. From the common commitment a, $\mathcal{E}xt$ gets C_{y_1}, C_{y_2}, C_{y_3}. As the three transcripts are accepting, $\mathcal{E}xt$ knows that (considering, w.l.o.g., $e_1 = 1, e_2 = 2, e_3 = 3$):

$$C_{y_1} = \mathsf{Com}(y_{e_1}, r_{y_{e_1}}) = \mathsf{Com}(y_{e_3+1}, r_{y_{e_3+1}}).$$
$$C_{y_2} = \mathsf{Com}(y_{e_2}, r_{y_{e_2}}) = \mathsf{Com}(y_{e_1+1}, r_{y_{e_1+1}}).$$
$$C_{y_3} = \mathsf{Com}(y_{e_3}, r_{y_{e_3}}) = \mathsf{Com}(y_{e_2+1}, r_{y_{e_2+1}}).$$

Then, if one of this equality verifies with different openings, then, due to the equivocability of the commitment scheme, $\mathcal{E}xt$ can extract the trapdoor. Then he can consider any value $\tilde{x}$, compute $\tilde{y} = f(\tilde{x})$ and compute the appropriate randomness to open C_y to $\tilde{y}$.

Now we consider the case when the equalities on the commitments traduce equalities of the openings. $\mathcal{E}xt$ thus obtains three values $y_1 = y_{e_1} = y_{e_3+1}$, $y_2 = y_{e_2} = y_{e_1+1}$, $y_3 = y_{e_3} = y_{e_2+1}$ and a single $y^* = y_1 \oplus y_2 \oplus y_3$. From then, as in the original ZKBoo proof he can execute back the MPC protocol and obtain three shares $x_1 = x_{e_1} = x_{e_3+1}$, $x_2 = x_{e_2} = x_{e_1+1}$, $x_3 = x_{e_3} = x_{e_2+1}$ and a single $x^* = x_1 \oplus x_2 \oplus x_3$ such that $y^* = f(x^*)$.

Now $\mathcal{E}xt$ needs to extract a randomness r^* that opens C_y to y^*. Using as a subroutine the extractors for the proofs Π_j, $\mathcal{E}xt$ obtains couples $(y'[j], r_{y'[j]})$ for $j \in [0, |y| - 1]$. From the protocol, as the transcripts are accepting ones,

$\mathcal{E}xt$ knows that $C_y = \sum_{i=0}^{|y'|-1} 2^i \mathsf{Com}(y'[i], r_{y'[i]})$. $\mathcal{E}xt$ selects one transcript, for instance e_1. He computes $\beta = y_{e_1} \oplus y_{e_1+1}$ and $C_z = \sum_{i=0}^{|y'|-1} 2^i C_{y'[i]\oplus\beta[i]} = \sum_{i=0}^{|y'|-1} 2^i \mathsf{Com}(y'[i] \oplus \beta[i], (-1)^{\beta[i]} r_{y'[i]})$. By the protocol, $C_z = C_{y_{e_1+2}} - \mathsf{Com}(0, r_z)$. If $\sum_{i=0}^{|y'|-1} 2^i (y'[i] \oplus \beta[i]) \neq y_{e_1+2}$ and/or $\sum_{i=0}^{|y'|-1} 2^i (-1)^{\beta[i]} r_{y'[i]} \neq r_{y_{e_1}+2} - r_z$, then again, $\mathcal{E}xt$ obtains the trapdoor of the commitment scheme and can open C_y to the value he wishes. Otherwise $\sum_{i=0}^{|y'|-1} 2^i (y'[i] \oplus \beta[i]) = y_{e_1+2}$ and $\sum_{i=0}^{|y'|-1} 2^i (y'[i]) = y_{e_1+2} \oplus \beta = y_{e_1+2} \oplus y_{e_1} \oplus y_{e_1+1} = f(x^*)$. Finally, $\sum_{i=0}^{|y'|-1} 2^i r_{y'[i]}$ opens C_y to $f(x^*)$ and the extractor is done. The running time of the extractor is bounded by the time of running back the MPC protocol (as for the ZKBoo extractor) + the running time of the extractors $\mathcal{E}xt_{\Pi_j}$ + computing one XOR and one commitment. Considering that an extractor for ZKBoo and the extractor for the proofs Π_j run in polynomial time, $\mathcal{E}xt$ also runs in polynomial time.

Zero-Knowledge. We consider a simulator $\mathcal{S}im$ that, on input a public statement C_y, shall produce a transcript (a, e, p). As for the soundness, we cannot call directly the ZKBoo simulator, $\mathcal{S}im_{ZKB}$, as the output of the circuit is not part of the statement. $\mathcal{S}im$ runs as follows: he sets e and he samples random tapes k_e, k_{e+1} and random input shares x_e, x_{e+1}. Then he runs the protocol as normal except that, when he meets a binary multiplication gate in the circuit, he cannot compute the real value of the view w_{e+1} (because it would depend on the third view that he cannot compute because he does not know x) so he samples it at random. This is indistinguishable form the real execution as binary multiplication gates are, in a correct execution, randomized with an element from k_{e+2} that the *Verifier* cannot compute. $\mathcal{S}im$ obtains output values y_e, y_{e+1}. He samples random r_e, r_{e+1}, computes $C_{y_e} = \mathsf{Com}(y_e, r_e)$ and $C_{y_{e+1}} = \mathsf{Com}(y_{e+1}, r_{e+1})$.

In a second step, he samples random $|y|-1$ bit values $y[j]$, $j \in [1, |y|-1]$ and associated randomness $r_y[j]$ and computes $C_{y[j]} = \mathsf{Com}(y[j], r[j])$. He executes the proofs Π_j with challenge e. Then he evaluates $C_y[0] = C_y - \sum_{j=1}^{|y|-1} C_{y[j]}$. Using the simulator for the proof Π_0, $\mathcal{S}im$ obtains a transcript for Π_0 for a challenge e'. If $e' \neq e$ he runs $\mathcal{S}im_{\Pi_0}$ again. Given that Π_0 is honest verifier, there is a non negligible probability that $e' = e$ within a polynomial time. Defining $\beta = y_e \oplus y_{e+1}$, $\mathcal{S}im$ can compute $C_{y[i]\oplus\beta[i]}$ only from the knowledge of $\mathcal{C}_{y[i]}$ and $\beta[i]$. Now $\mathcal{S}im$ samples $r_z \in \mathbb{Z}_p$ and computes $C_{y_{e+2}} = \sum_{j=0}^{|y|-1} C_{y[i]\oplus\beta[i]} + \mathsf{Com}(0, r_z)$. He now has all the elements to produce an accepting transcript.

The transcript of the ZKBoo part of the proof is indistinguishable form a real execution. The elements that $\mathcal{S}im$ produces itself are commitments that will not be opened, hence, by the hiding property of the commitment, the complete simulated transcript is indistinguishable form a real execution of the protocol.

4.2 A Second Solution: `CopraZK`

Let us denote by $\mathsf{Func}(\mathcal{D}, \mathcal{R})$ the set of all functions from $\mathcal{D}$ to $\mathcal{R}$ and by $\mathsf{FF}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ the set of all function families with parameter (key) in $\mathcal{K}$, domain $\mathcal{D}$ and range $\mathcal{R}$. We write $f : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ for a function family in $\mathsf{FF}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ (and

Let C_y, C_x be public commitments and m a public message. The *Prover* wants to convince the *Verifier* that he knows x, r_x, r_y such that $C_x = \mathsf{Com}(x, r_x)$ and $C_y = \mathsf{Com}(f(x,m), r_y)$.

Prover

Commit phase:

1. computes $\alpha = \mathsf{h}(C_x || C_y)$.
2. computes $t = f(x,m) + \alpha x \mod q$.
3. Evaluates the commit phase output a_Π for the Σ protocol $\Pi = PK\{x, r_x, y, r_y : C_x = xP + r_xQ \wedge C_y = yP + r_yQ \wedge t = y + \alpha x\}$

for $\rho \in [1, \sigma]$ (ZKBoo part):

4. samples random tapes $k_1^\rho, k_2^\rho, k_3^\rho$.
5. generates the shares $x_1^\rho, x_2^\rho, x_3^\rho = \mathsf{Share}(x, k_1^\rho, k_2^\rho)$ such that $x = x_1^\rho \oplus x_2^\rho \oplus x_3^\rho$.
6. evaluates the MPC protocol on the circuit Circ that, on entrance values entry (x, a), evaluates $t = f(x,m) + \alpha x \mod q$ and obtains three views $w_1^\rho, w_2^\rho, w_3^\rho$.
7. obtains the output shares : $o_1^\rho = t_1, o_2^\rho = t_2, o_3^\rho = t_3$ such that $t = t_1^\rho \oplus t_2^\rho \oplus t_3^\rho$.
8. commits to the views : $c_1^\rho = \mathsf{h}(w_1^\rho, k_1^\rho), c_2^\rho = \mathsf{h}(w_2^\rho, k_2^\rho), c_3^\rho = \mathsf{h}(w_3^\rho, k_3^\rho)$.

$a = t, (c_1^\rho, c_2^\rho, c_3^\rho, o^\rho = (o_1^\rho, o_2^\rho, o_3^\rho))_\rho, a_\Pi$.

Challenge: e = h(a)

Response phase:

1. computes the response z_Π for the proof Π

for $\rho \in [1, \sigma]$:

2. $b^\rho = (o_{e+2}^\rho = t_{e+2}, c_{e+2}^\rho)$
3. $z^\rho = (w_{e+1}^\rho, k_e^\rho, k_{e+1}^\rho)$

returns $p = (e, (b^\rho, z^\rho)_\rho, z_\Pi)$

. . . . . .

Verifier(a, p)

1. parses p as e, $(b^\rho, z^\rho)_\rho, z_\Pi$
2. parses a as $t, (c_1^\rho, c_2^\rho, c_3^\rho, o^\rho = (o_1^\rho, o_2^\rho, o_3^\rho))_\rho, a_\Pi$
3. computes α
4. reconstruct the proof Π

for $\rho \in [1, \sigma]$ (Verifying the ZKBoo proof):

5. runs the MPC protocol to reconstruct w_e^ρ from $w_{e+1}^\rho, k_e^\rho, k_{e+1}^\rho$
6. obtains $t_e^\rho = \mathsf{Output}(w_e)$, $t_{e+1}^\rho = \mathsf{Output}(w_{e+1})$
7. computes $t_{e+2}^\rho = t \oplus t_e^\rho \oplus t_{e+1}^\rho$.

reconstructs a and rejects if $e \neq h(a)$

Fig. 4. Our protocol `CopraZK`.

call it a function, by ease of language). Let f be a function: $\mathbb{Z}_2^\ell \times \mathbb{Z}_2^* \to \mathbb{Z}_2^\ell$ and m a public input, $m \in \mathbb{Z}_2^*$. Let $\mathbb{G}$ be a group of prime order q, such that $2^\ell \leq q$. Let P be a generator for this group and Q an element of $\mathbb{G}$ such that $\log_P(Q)$ is unknown. Let h be a hash function $\mathbb{Z}_2^* \to \mathbb{Z}_2^{\ell *}$ and Com be the Pedersen commitment scheme. They are the public parameters of the protocol.

Let C_x, C_y be public commitments, known to the verifier. The main idea is to consider the circuit that computes the tag $t = f(x, m) + \alpha x$ where α is a public coefficient derived from the commitment values C_x and C_y. A MPC in the head proof on this circuit ensures that t is correctly computed from a secret value x known to the prover. Considering Pedersen commitments, we complete the circuit proof with an algebraic proof that the committed values in C_x and C_y verify the relation t. This linear relation plus the properties of f defined below, bind the values of C_x and C_y such that the verifier can be convinced that the value committed in C_y is equal to the evaluation of f on the value committed in C_x. A complete description is given in Fig. 4. We depict our protocol using ZKBoo for the circuit part, but the proof adapts to any circuit based ZK proof.

Theorem 2 states the security of `CopraZK`, that is settled on two properties of the family f. The correlation intractability [15] of its dual function $\tilde{f}$ (when the role of the input and the key are switched) and a glider-PRF (general linear input deviation resistant PRF) security that states that an adversary gains no knowledge on x when given $f(x, m) + \alpha x$. We provide more information on those properties and a sketch of proof in Appendix B.

Theorem 2. *Given that ZKBoo and the Π_j are Σ protocols with 3 (respectively 2) special soundness and honest verifier Zero Knowledge property (in their interactive form), that* Com *is a homomorphic commitment scheme, and that f is a* glider-PRF *function family such that $\tilde{f}$ is correlation intractable relatively to relations $\{\mathrm{R}_{a,b} : \{x, y : y = ax + b\}\}$, then the protocol described in Fig. 4 is a Σ-protocol with 3-special soundness and full Zero-Knowledge.*

5 Conclusion

In this work we provide a concrete solution to a practical problem that appears in the MLS specification. We describe how existing cryptographic tools such as ZK proofs and verifiable encryption can be combined to secure the update process. As the regular update of the group secret is the key to obtain the FS and PCS properties, we think our solution may be of interest.

Additionally, we propose two protocols to obtain ZK proofs on circuit with committed input and output, such that our improvement proposal for MLS is settled on protocols as efficient as possible. Hence, an interesting way for future work is in the optimization of the verifiable encryption. The CL framework, introduced by Castagnos and Laguillaumie in [17] and enriched with Zero-Knowledge properties in [16], that considers a cyclic group $\mathbb{G}$ where the DDH assumption holds together with a subgroup F of $\mathbb{G}$ where the discrete logarithm problem is easy, may provide novel and efficient solutions.

A Key Size and Group Orders in MLS Updates

In [7], several suitable cipher suites are described. We focus on one of them for a practical example, for a 128-bit security level. This suite uses X25519

for ECDH computation and SHA256 as a hash function (and base function for HKDF implementation). Following [24], the private key sk is obtained from a 256-bit string of secure random data $(sk[0], sk[1], \ldots, sk[255])$ by applying the following transform: $sk[0]\& = 248$, $sk[31]\& = 127$ and $sk[31]| = 64$. One obtains, when interpreted as an integer value in little endian, a scalar of the form $2^{254} + 8 \cdot \ell, \ell \in [\![0; 2^{251} - 1]\!]$. We design by $\mathsf{deriveSK}$ the application of $\mathsf{SHA256}$ followed by the above transformation such that for any 32-byte sequence of random data X, $\mathsf{deriveSK}(\mathsf{X})$ is a valid secret key for X25519. This encoding can be integrated in the circuit computing the last derivation. The public key is obtained by multiplying the secret key by the base point of the curve: given a 32-byte secret X, $\mathsf{DeriveKeyPair}(X) = (\mathsf{deriveSK}(X), \mathsf{deriveSK}(X)P)$. We adopt this notation independently from the curve targeted.

Group Order and Commitments. In our proofs, we consider commitments and discrete logarithm proofs in cyclic groups of order q, and circuits input and output that naturally lie in F_q. This may not be the case. Considering X25519 key derivation derived above, a new user's secret ps'_B is a random element in $\{0,1\}^{256}$, which, when interpreted as an integer, can be larger than q. As explained in [6], it is possible to consider $ps'_B \mod q$ for the commitment and to include to a modular computation in the circuit. If q is close enough to 2^{256} then it is a simple comparison and subtraction. This requires around 2 000 gates, which is negligible compared to our circuit size. Another solution is to directly sample ps'_B in F_q. This can be done by rejection sampling or as follow: sample X sufficiently big compared to $\log_2(q)$ ($\log_2(X) > \log_2(q) + 64$ as advised by the NIST for instance), then simply considering $X \mod q$ can be done with a negligible bias. For all the intermediate values in the tree, the first method can be applied. The last step is the commitment of the secret key $sk = \mathsf{Encode}(X)$. For this element, we directly consider the encoding provided with the curve. The commitment C_{sk} of sk in a group of order q will result in the same implicit reduction modulo q than the computation of the public key. Then we can produce an AND ZK proof that the value committed to in C_{sk} is the discrete log of the pk: $PK\{sk : C_{sk} = skP + rQ \wedge pk = skP\}$.

B Security of Our Zero-Knowledge Protocols

We present a sketch of proof for the security of `CopraZK` given in Theorem 2. It is settled on two properties for the function family f. Firstly, we need its dual function $\tilde{f}$ to be correlation intractable with respect to the family of relations $\mathcal{R}_{a,b} : \{x, y : y = ax + b\}$ for a, b random values. Correlation intractability was introduced in [15] and says that, for any relation in the family $\mathcal{R}_{a,b}$, for any random key x, an adversary has a negligible probability to find an input m such that $(m, f(x, m))$ satisfies the relation. Secondly, we need to be sure that the tag does not leak information on the key. We define a general linear input deviation resistant PRF ($\mathsf{glider\text{-}PRF}$) as follows:

Definition 1 (glider-PRF **security).** *A function family* $f \in \mathsf{FF}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ *(with appropriate domain and range) is said to be a* glider-PRF *if for all PPT adversary* $\mathcal{A}$*, and a random* $\alpha \leftarrow_{\$} \mathcal{R}$*, there exists a negligible function* negl *such that:*

$$\Pr\left[x \leftarrow \mathcal{A}(\alpha)^{\mathcal{O}_g} : g \leftarrow_{\$} \mathsf{F}(\mathcal{D}, \mathcal{R})\right] - \Pr\left[x \leftarrow \mathcal{A}(\alpha)^{\mathcal{O}_{f(k,\cdot)+\alpha\cdot}} : k \leftarrow_{\$} \mathcal{K}\right] \leq \mathsf{negl}(\lambda)$$

As we use Fiat-Shamir to get an non interactive protocol, our proof is settled in the Random Oracle Model (ROM), which would satisfy our hypothesis. However, it seems contradictory to idealize as a random oracle the PRF f that is concretely described as a circuit in the ZKBoo part of the protocol. Hence, the ROM hypothesis only applies to the hash function h that generates the challenge and the correlation intractability and glider-PRF properties provide a way to formalize a security proof when only some properties of the random oracle are needed. The correctness of the protocol follows by inspection.

3-Special Soundness. From the 2-special soundness of the Sigma protocol Π, one extract $\tilde{x}, \tilde{y}, \tilde{r_x}, \tilde{r_y}$ such that $C_x = \tilde{x}P + \tilde{r_x}Q, C_y = \tilde{y}P + \tilde{r_y}Q$ and $t = \alpha\tilde{x} + \tilde{y}$. From the 3-special soundness of ZKBoo, one extracts x' such that $t = f(x', m) + \alpha x'$, where t is a fixed value, the same as the one for the proof Π. Correlation intractability of $\tilde{f}$ ensures that $x' \neq \tilde{x}$ happens with negligible probability. We note that the correlation intractability of $\tilde{f}$ requires the input of f to be randomized. In MLS, this supposes considering a random value (for instance the hash of the tree view) instead of the constant 0.

Zero Knowledge. We build a simulator $\mathcal{S}im$ as follows: $\mathcal{S}im$ sample a random value $t \leftarrow_{\$} \mathbb{Z}/q\mathbb{Z}$. Then he calls the Simulator of ZKBoo, $\mathcal{S}im_{ZKBoo}$, as a subroutine and obtains a transcript $(a_{ZKBoo,e,z_{ZKBoo}})$. Then he calls the simulator for the Sigma protocol Π, $\mathcal{S}im_{\Pi}$, as a second subroutine, on the challenge e and obtains a second transcript (a_{Π}, e, z_{Π}) (as $\mathcal{S}im_{\Pi}$ shall work for any challenge). If f is glider-PRF-secure, then sampling a random t is indistinguishable from the real distribution of t and finally, the output distribution of $\mathcal{S}im$ is indistinguishable from the real execution output. In the context of MLS, the tag t must be accessible to the server only. A user who would receive its valid update and access the tag could compute the secret of its child, which he should not.

References

1. Agrawal, S., Ganesh, C., Mohassel, P.: Non-interactive zero-knowledge proofs for composite statements. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 643–673. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_22
2. Alwen, J., et al.: Keep the dirt: tainted TreeKEM, adaptively and actively secure continuous group key agreement. Cryptology ePrint Archive, Report 2019/1489 (2019). https://eprint.iacr.org/2019/1489
3. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: security notions, proofs, and modularization for the signal protocol. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 129–158. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_5

4. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 248–277. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_9
5. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 2087–2104. ACM Press, Oct/Nov 2017. https://doi.org/10.1145/3133956.3134104
6. Backes, M., Hanzlik, L., Herzberg, A., Kate, A., Pryvalov, I.: Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 286–313. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_10
7. Barnes, R., Beurdouche, B., Millican, J., Omara, E., Cohn-Gordon, K., Robert, R.: The messaging layer security (MLS) protocol. https://datatracker.ietf.org/doc/draft-ietf-mls-protocol/
8. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.: Hybrid public key encryption (2021). https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-hpke-12
9. Bellare, M., Singh, A.C., Jaeger, J., Nyayapati, M., Stepanovs, I.: Ratcheted encryption and key exchange: the security of messaging. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 619–650. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_21
10. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018). https://eprint.iacr.org/2018/046
11. Bhargavan, K., Barnes, R., Rescorla, E.: TreeKEM: asynchronous decentralized key management for large dynamic groups (2018)
12. Brzuska, C., Cornelissen, E., Kohbrok, K.: Cryptographic security of the MLS RFC, Draft 11. Cryptology ePrint Archive, Report 2021/137 (2021). https://ia.cr/2021/137
13. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_25
14. Camenish, J., Stadler, M.: Proof systems for general statements about discrete logarithms (1997)
15. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Cryptology ePrint Archive, Report 1998/011 (1998). http://eprint.iacr.org/1998/011
16. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 191–221. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_7
17. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_26
18. Chase, M., et al.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 1825–1842. ACM Press, Oct/Nov 2017. https://doi.org/10.1145/3133956.3133997

19. Chase, M., Ganesh, C., Mohassel, P.: Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 499–530. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_18
20. Chase, M., Perrin, T., Zaverucha, G.: The signal private group system and anonymous credentials supporting efficient verifiable encryption. Cryptology ePrint Archive, Report 2019/1416 (2019). https://eprint.iacr.org/2019/1416
21. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: asynchronous group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2017/666 (2017). http://eprint.iacr.org/2017/666
22. Cohn-Gordon, K., Cremers, C.J.F., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, 26–28 April 2017, pp. 451–466 (2017). https://doi.org/10.1007/s00145-020-09360-1
23. Damgård, I.: On sigma protocols (2010)
24. Bernstein, D.J.: A state-of-the-art Diffie Hellman function. https://cr.yp.to/ecdh.html
25. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
26. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
27. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: faster zero-knowledge for Boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016, pp. 1069–1083. USENIX Association, August 2016
28. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_19
29. Gvili, Y., Ha, J., Scheffler, S., Varia, M., Yang, Z., Zhang, X.: TurboIKOS: improved non-interactive zero knowledge and post-quantum signatures. Cryptology ePrint Archive, Report 2021/478 (2021). https://eprint.iacr.org/2021/478
30. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC, pp. 21–30. ACM Press, June 2007. https://doi.org/10.1145/1250790.1250794
31. Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: the safety of messaging. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_2
32. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 955–966. ACM Press, November 2013. https://doi.org/10.1145/2508859.2516662
33. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: almost-optimal guarantees for secure messaging. Cryptology ePrint Archive, Report 2018/954 (2018). https://eprint.iacr.org/2018/954

34. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 525–537. ACM Press, October 2018. https://doi.org/10.1145/3243734.3243805
35. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC, pp. 723–732. ACM Press, May 1992. https://doi.org/10.1145/129712.129782
36. Krawczyk, H.: Cryptographic extraction and key derivation: the HKDF scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_34
37. Marlinspike, M., Perrin, T.: The double ratchet algorithm. Signal's web site (2016)
38. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
39. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 3–32. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_1
40. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_17