Short Concurrent Covert Authenticated Key Exchange (Short cAKE)

Karim Eldefrawy¹, Nicholas Genise^{2*}, and Stanislaw Jarecki³

¹ SRI International, karim.eldefrawy@sri.com

 $^2\,$ Duality Technologies, ngenise@dualitytech.com

³ University of California, Irvine, sjarecki@uci.edu

Abstract. Von Ahn, Hopper and Langford introduced the notion of steganographic a.k.a. covert computation, to capture distributed computation where the attackers must not be able to distinguish honest parties from entities emitting random bitstrings. This indistinguishability should hold for the duration of the computation except for what is revealed by the intended outputs of the computed functionality. An important case of covert computation is mutually authenticated key exchange, a.k.a. mutual authentication. Mutual authentication is a fundamental primitive often preceding more complex secure protocols used for distributed computation. However, standard authentication implementations are not covert, which allows a network adversary to target or block parties who engage in authentication. Therefore, mutual authentication is one of the premier use cases of covert computation and has numerous real-world applications, e.g., for enabling authentication over steganographic channels in a network controlled by a discriminatory entity.

We improve on the state of the art in covert authentication by presenting a protocol that retains covertness and security under *concurrent composition*, has minimal message complexity, and reduces protocol bandwidth by an order of magnitude compared to previous constructions. To model the security of our scheme we develop a UC model which captures standard features of secure mutual authentication but extends them to covertness. We prove our construction secure in this UC model. We also provide a proof-of-concept implementation of our scheme.

1 Introduction

Steganography in the context of secure computation deals with hiding executions of secure computation protocols.⁴ Such hiding is only possible if the participating parties have access to (public) communication channels which are *steganographic*, i.e., which naturally exhibit some entropy. Cryptographic protocols over such channels can be steganographic, a.k.a. *covert*, if all protocol messages the protocol exchanges cannot be distinguished from (assumed) a priori random behavior of the communication channels.

The study of covert secure computation was initiated by Hopper et al. [38] for the two-party case, and by Chandran et al. [21] and Goyal and Jain [35] for the multi-party case. Both [38] and [21,35] prove feasibility for covert computation of arbitrary circuits which tolerates passive and malicious adversaries, respectively. Subsequently, Jarecki [40] showed that general maliciously-secure two-party covert computation can be roughly as efficient as standard, i.e., non-covert, secure computation.

A flagship covert computation application is *covert authentication* and covert Authenticated Key Exchange (cAKE). In a cAKE protocol, two parties can authenticate each other as holders of mutually accepted certificates, but an entity who does not hold proper certificates, in addition to being unable to authenticate, cannot even distinguish a party that executes a covert AKE from a random beacon, i.e., from noise on the steganographic channel. In essence, cAKE allows group members to authenticate one another, but their presence on any steganographic communication channel is *entirely hidden*, i.e., they are invisible.

The application of covert computation to covert AKE has been addressed by Jarecki [39], but the state of the art in covert AKE is significantly lacking in several aspects: large bandwidth, high round complexity, and (a lack of) security under concurrent composition. Regarding security, the scheme of [39] achieves only sequential security, and does not ensure independence of keys across sessions, which is insufficient for full-fledged (covert) AKE.⁵ Regarding round complexity and bandwidth, the cAKE protocol in [39] requires 6 message flows and relies on a composite-order group (and a factoring assumption),

^{*} This work was done while the second author was at SRI International.

 $^{^4}$ This is a full version of a work published in [28].

⁵ In particular, [39] does not imply security against man in the middle attacks.

resulting in bandwidth which can be estimated as at least 3.6kB. Recent works on random encodings of elliptic curve points, e.g. [11,59], allow for potentially dramatic bandwidth reduction if secure cAKE can be instantiated over a prime-order group.

Covert vs. Standard Authentication. Covert Authenticated Key Exchange (cAKE) can be formalized as a secure realization of functionality $\mathcal{F}_{cAKE}[C]$ shown in Figure 1's entirety, characterized by a given *admission function* C. Let us first set the terms by explaining the standard, i.e. non-covert, AKE functionality $\mathcal{F}_{AKE}[C, L]$, characterized by C and a *leakage function* L, which is portrayed in the same figure. Reading Figure 1 with dashed text and without greyed text defines $\mathcal{F}_{AKE}[C, L]$, and with greyed text and without dashed text defines \mathcal{F}_{cAKE} .



Fig. 1. Standard AKE functionality $\mathcal{F}_{AKE}[C, L]$ includes dashed text & omits greyed text; Covert AKE functionality $\mathcal{F}_{cAKE}[C]$ includes greyed text & omits dashed text.

In an AKE protocol, i.e. a protocol that realizes \mathcal{F}_{AKE} , parties P_1 and P_2 run on inputs x_1 and x_2 , which represent their *authentication tokens*, e.g. passwords, certificates, keys, etc., and if these inputs match each other's admission policy, jointly represented by circuit C, then P_1 and P_2 establish a shared random session key $K_1 = K_2$, otherwise their outputs K_1, K_2 are independent.⁶ If L is a non-trivial function, then the protocol leaks L(x) on P's input x to P's counterparty.

For example, Password Authenticated Key Exchange (PAKE) [7] can be defined as (secure realization of) $\mathcal{F}_{AKE}[C_{pa}]$ where C_{pa} is an equality test, i.e., $C_{pa}(x_1, x_2) = 1$ if and only if $x_1 = x_2$. In another example, a standard notion of AKE, e.g. [27], which we will call here as a *Fixed Public Key* AKE (FPK-AKE) to distinguish it from other AKE types, can be defined as $\mathcal{F}_{AKE}[C_{fpk}, L_{fpk}]$ where $C_{fpk}(x_1, x_2) = 1$ iff $x_1 = (sk_1, pk_2)$ and $x_2 = (sk_2, pk_1)$ s.t. pk_1, pk_2 are the public keys corresponding to resp. sk_1, sk_2 . Leakage L_{fpk} is typically omitted in the works on FPK-AKE, e.g. [3,20], because it is assumed that public keys pk_i of each P_i are public inputs. However, the implicit leakage profile in these works is $L_{fpk}((sk_P, pk_{CP})) = (pk_P, pk_{CP})$ where pk_P is a public key corresponding to sk_P .⁷

We say that protocol Auth UC-realizes a *covert* AKE functionality \mathcal{F}_{cAKE} if it does so under a constraint that a real-world party P invoked on input $x = \bot$ does not follow protocol Auth but instead emulates a *random beacon* Auth^{\$(\kappa)} defined as follows: In each round, if Auth participant sends an $n(\kappa)$ -bit message then Auth^{\$(\kappa)} sends out an $n(\kappa)$ -bit random bitstring, where κ is a security parameter. In more detail, a covert AKE functionality $\mathcal{F}_{cAKE}[C]$ makes the following changes to the standard AKE functionality $\mathcal{F}_{AKE}[C, L]$: First, \mathcal{F}_{cAKE} eliminates leakage L(x), equivalently $L(x) = \bot$ for all x. Second, \mathcal{F}_{cAKE} admits a special input $x = \bot$ which designates P as a random beacon, i.e., it tells P to run Auth^{\$(\kappa)} instead of Auth. Third, \mathcal{F}_{cAKE} adds the check that $x_1 \neq \bot$ and $x_2 \neq \bot$ to the condition for setting $K_1 = K_2$. Fourth, the functionality ensures that if P's input is \bot , i.e. P is a non-participant, then its output is \bot .

Implications of Covert AKE. The first impact of covert AKE vs. the standard AKE, is that if we disregard what P_1 does with its output key K_1 , then a malicious P_2^* cannot distinguish an interaction with a real party P_1 (where $x_1 \neq \bot$) and a random beacon (where $x_1 = \bot$) because in either case \mathcal{F}_{cAKE} gives P_2^* the same output, a random key K_2 . Indeed, the only way P_2^* can distinguish cAKE participant

⁶ Note that Figure 1 defines AKE as a key exchange without explicit entity authentication, but the latter can be added to any AKE by testing if parties output the same key via any key confirmation protocol.

⁷ In a standard FPK-AKE protocol party P can reveal either key. E.g. Sigma [44] used in TLS reveals P's own key pk_{P} , while SKEME [43] reveals key pk_{CP} which party P assumes for its counterparty, unless it employs key-private encryption [4].

 P_1 from a random beacon, is not the cAKE protocol itself, but an application which P_1 might run using cAKE's output K_1 . There are three cases of P_1 from P_2^* 's point of view, where x_2^* is P_2^* 's input to \mathcal{F}_{cAKE} :

- (1) P_1 = protocol party with x_1 s.t. $C(x_1, x_2^*) = 1$, in which case P_2^* learns K_1 ;
- (2) P_1 = protocol party with x_1 s.t. $\mathsf{C}(x_1, x_2^*) = 0$, in which case K_1 is hidden;
- (3) P_1 = random beacon, represented by $x_1 = \bot$, in which case $K_1 = \bot$.

The second property that cAKE adds to a standard AKE is that *if* the upper-layer application Π which P_1 runs on cAKE's output K_1 continues using steganographic channels, and P_1 encrypts Π 's messages on these channels under key K_1 , then P_2^* cannot distinguish cases (2) and (3). That is, P_2^* cannot tell a real-world P_1 who ran cAKE on inputs that didn't match x_2^* and then runs Π on cAKE output K_1 , from a random beacon.⁸ Detecting case (1) from a random beacon depends on the upper-layer protocol Π : If Π is non-covert than P_2^* will confirm that P_1 is a real-world party by running protocol Π on input K_1 (which P_2^* learns if $\mathsf{C}(x_1, x_2^*) = 1$). However, if protocol Π is itself covert then P_1 will continue to be indistinguishable from a random beacon even in case (1). In other words, cAKE protocols are *composable*, e.g. running a covert PIN-authenticated KE, encrypted by a key created by a covert PAKE, ensures covertness to anyone except a party who holds both the correct password and the PIN.

Group Covert AKE (Group cAKE). In this work we target a "group" variant of cAKE. Namely, P's authentication token is a pair x = (gpk, cert) where gpk is a public key identifying a group, cert is a certificate of membership in this group, and the admission function $C_G(x_1, x_2)$ outputs 1 if and only if $\exists gpk$ s.t. $x_1 = (gpk, cert_1), x_2 = (gpk, cert_2),$ and $Ver(gpk, cert_1) = Ver(gpk, cert_2) = 1$, where Ver stands for certificate verification. In other words, both parties must assume the same group identified by gpkand each must hold a valid membership certificate in this group. We assume that key gpk is generated by a trusted group manager together with a master secret key msk which is used to issue valid certificates, and that the certification scheme is unforgeable, i.e. that an adversary which sees any number of valid certificates $cert_1, ..., cert_n$ cannot output $cert^*$ s.t. $Ver(gpk, cert^*) = 1$ and $\forall i \ cert^* \neq cert_i$.

The above setting of group cAKE is the same as that of group signatures [22], except that membership certificates are used to authenticate, not to sign,⁹ and the authentication is covert. However, note that a straightforward usage of group signatures for authentication, e.g. where two parties sign a key exchange transcript using group signatures, can at best realize $\mathcal{F}_{AKE}[C_G, L]$ where leakage L hides P_i 's certificate (and hence P_i 's identity) but reveals the group public key gpk, because a group signature is verifiable under this key.¹⁰

In practice, a certification scheme must admit *revocation*, i.e. a group manager must be able to revoke a certificate, e.g. by distributing revocation token rt s.t. (1) there is an efficient procedure Link which links a certificate to this token, i.e. if Ver(gpk, cert) = 1 then Link(cert, rt) = 1 for rt associated with *cert*, and (2) certificates remain unforgeable in the presence of revocation tokens.¹¹ If Link(cert, RTset) stands for a procedure which outputs 1 iff $\exists rt \in RTset$ s.t. Link(cert, rt) = 1, then we define group covert AKE(with revocation), or simply group cAKE, as $\mathcal{F}_{cAKE}[C_{Gwr}]$ where $C_{Gwr}(x_1, x_2) = 1$ iff

- 1. $\exists gpk \text{ s.t. } x_1 = (gpk, cert_1, \mathsf{RTset}_1) \text{ and } x_2 = (gpk, cert_2, \mathsf{RTset}_2),$
- 2. $\operatorname{Ver}(gpk, cert_1) = \operatorname{Ver}(gpk, cert_2) = 1$,
- 3. and $\text{Link}(cert_2, \text{RTset}_1) = \text{Link}(cert_1, \text{RTset}_2) = 0$.

In other words, parties establish a shared secret key if both assume the same group public key, both hold valid certificates under this key, and neither certificate is revoked by the revocation information held by a counterparty.

Applications of group cAKE. Authentication and key exchange are fundamental primitives that regularly precede secure protocols used for distributed online computations. Identifying executions of such protocols is often used as a first step when blocking communication [54] or targeting it for filtering

⁸ This requires encryption with ciphertexts indistinguishable from random bitstrings, but this is achieved by standard block cipher modes, CBC, OFB, or RND-CTR.

⁹ Using group signatures for authentication is known as an *Identity Escrow* [42].

¹⁰ Secret Handshake [2] flips this leakage, realizing $\mathcal{F}_{AKE}[C_G, L']$ for L' that hides gpk but reveals a one-way function of P_i 's certificate. To complete comparisons, standard PKI-based AKE realizes $\mathcal{F}_{AKE}[C_G, L'']$ s.t. L'' reveals both a root of trust gpk and a one-way function of P_i 's certificate, namely P_i 's public key with gpk's signature.

¹¹ Here we follow the *verifier-local revocation* model [13], but other models are possible, e.g. using cryptographic *accumulators* [8,17].

or other attacks [57,60]. Authentication is thus a natural primitive to be protected and rendered covert to avoid such blocking or targeting. To the best of our knowledge, there are currently no practical covert AKE protocols implemented, let alone deployed in distributed systems. If they existed, such protocols could help hide and protect communication required for authentication and key establishment in such systems. Since our work demonstrates that covert authentication can be realized with a (computation and communication) cost very close to that required for existing non-covert anonymous authentication (e.g., anonymous credentials [16]) or indeed standard non-private authentication (e.g., TLS handshake with certificate-based authentication), we argue that such protocols could become an enabling tool in large-scale resilient anonymous communication systems. Such anonymous communication systems have been the focus of the recent DARPA research program on developing a distributed system for Resilient Anonymous, end-to-end mobile communication that would be attack-resilient and reside entirely within a contested network environment," and its targets included stenographic hiding of communication participants [55]. An efficient covert authentication could be play an essential role in such a system.

Other Variants of Covert AKE. There are other natural variants of covert AKE which can be implemented using known techniques, but none of them imply a practical group cAKE. Covert PAKE corresponds to $\mathcal{F}_{cAKE}[C_{pa}]$, for C_{pa} defined above. Several known efficient PAKE schemes, e.g. EKE [7] and SPAKE2 [1], most likely realize $\mathcal{F}_{cAKE}[C_{pa}]$ after simple implementation adjustments, e.g. SPAKE2 should use an elliptic curve with a *uniform encoding*, which maps a random curve point to a random fixedlength bitstring, see Section 2.4. (We believe this is likely to hold because these PAKE protocols exchanges random group elements, or ideal-cipher encryptions of such elements.) The covert Fixed Public Key AKE (FPK-AKE) corresponds to $\mathcal{F}_{cAKE}[C_{fpk}]$, for C_{fpk} defined above. The work on key-hiding AKE [36] shows that several FPK-AKE protocols, namely 3DH [49], HMQV [45], and SKEME [43] instantiated with keyprivate and PCA-secure encryption, realize $\mathcal{F}_{AKE}[C_{fpk}]$, i.e. FPK-AKE without leakage, and after similar implementation adjustments as in the case of SPAKE2, these protocols probably realize $\mathcal{F}_{cAKE}[C_{fpk}]$. (This is likely to hold for similar reason, because these FPK-AKE protocols exchange random group elements and ciphertexts.) Another variant is an *identity based* AKE (IB-AKE), where public key pk is replaced by an identity and gpk is a public key of a Key Distribution Center. Covert IB-AKE can be implemented using Identity-Based Encryption (IBE) with covertly encodeable ciphertexts, such as the Boneh-Franklin IBE [12] given a bilinear map group with a covert encoding.

However, it is unclear how to efficiently implement group cAKE from covert PAKE, FPK-AKE, or IB-AKE. Using any of these tools each group member would have to hold a separate token for every other group member (be it a password, a public key, or an identity), and the authentication protocol would need to involve n parallel instances of the covert PAKE/FPK-AKE/IB-AKE. Using the multiplexing technique of [48,23] such parallel execution can be done covertly at $\tilde{O}(n)$ cost, but this would not scale well. Either of these $\tilde{O}(n)$ -cost implementations can be seen as implementing a *covert Broadcast Encryption (BE)* with O(n)-sized ciphertext. Indeed, any *covert* broadcast encryption implies cAKE. However, even though there are broadcast encryption schemes with sublinear ciphertexts, e.g. [29], to the best of our knowledge there are no sublinear BE schemes which are key-private [4], let alone covert.

1.1 Our Contributions

We show the *first practical covert group cAKE scheme*, with support for certificate revocation, with the following features:

- 1. Universally composable (UC) covertness and security: We formalize a universally composable (UC) [19] functionality for group cAKE, and show a scheme which realizes it. In particular, this implies that our group cAKE scheme retains covertness and security under concurrent composition, and that each session outputs an independent key, as expected of a secure AKE.
- 2. Practically efficient: Our group cAKE scheme is round minimal, using one simultaneous flow from each party, and bandwidth efficient, with a message size of four DDH group elements and two points in a type-3 bilinear curve, resulting in bandwidth of 351B, factor of 10x improvement over state of the art. Our group cAKE scheme also has a low computational overhead of 14 exponentiations and 4 + n bilinear maps per party, where n is the size of the revocation list. Note that these parameters are a constant factor away from non-covert Group AKE, or indeed any other (A)KE. (The most significant slowdown compared to standard AKE comes from using bilinear maps.)

Furthermore, the above security and round improvements are enabled by security improvements in a crucial tool used in covert computation, namely a covert *Conditional Key Encapsulation Mechanism* (CKEM) [21,39],¹² which we construct for any language with so-called Sigma-protocol, i.e. a 3-round public-coin honest-verifier zero-knowledge proof of knowledge [26]. Covert CKEM is a covert KEM version of Witness Encryption [32]: It allows the sender to encrypt a key under a statement x, where decryption requires knowledge of a witness w for membership of statement x in a language \mathcal{L} chosen at encryption. This KEM is covert if the ciphertext is indistinguishable from a random string, and in particular cannot be linked to either language \mathcal{L} or statement x. The security improvements in covert CKEM are of independent interest because covert CKEM is a covert counterpart of a zero-knowledge proof, and as such it is a general-purpose tool which can find applications in other protocols.

Technical Overview. The high-level idea of our group cAKE construction follows the blueprint used for group cAKE by Jarecki [39]. Namely, it constructs group cAKE generically from a covert Identity Escrow (IE) scheme [42] and a covert CKEM: Each party sends a (covert) commitment to its IE certificate to the counterparty, and each party runs a CKEM, once as the sender (S) and once as the receiver (R), where the latter is proving ownership and validity of the committed certificate. Each party runs the CKEM once as the receiver and once as the sender, since the protocol covertly computes an AND statement: given (gpk, cert) from P and (gpk', cert') from P', it checks that $(cert \in \mathcal{L}^{\mathsf{IE}}(gpk')) \land (cert' \in \mathcal{L}^{\mathsf{IE}}(gpk))$ where $\mathcal{L}^{\mathsf{IE}}(gpk)$ is the language of valid IE certificates generated under gpk. Finally, each party checks the received committed certificate against their revocation list.¹³ If the revocation check passes, each party uses the two CKEM outputs to derive a session key.

The main technical challenge is constructing provable secure group cAKE which is universally composable. To achieve this we implement several significant upgrades to the covert CKEM notion defined and constructed in [39] (for the same general class of languages with Sigma-protocols):

(1) First, we combine strong soundness of [39] and simulation-soundness of [9] to strong simulationsoundness. I.e., we require an efficient extractor that extracts a witness from an attacker who distinguishes S's output key from random on instance x in the presence of a simulator which plays R role on any instance $x' \neq x$. Strong simulation-soundness is needed in a concurrent group cAKE to let the reduction extract a certificate forgery from an attacker who decrypts a covert CKEM on a statement corresponding to a non-revoked certificate, while the reduction simulates all CKEM's on behalf of honest R's.

(2) Second, we amend covert CKEM with a *postponed-statement zero-knowledge* property, i.e. we require a postponed-statement simulator for simulating the CKEM on behalf of a receiver R. Such simulator must compute the same key an honest R would compute, and do so not only without knowing R's witness but also *without knowing the statement* used by R, until after all covert CKEM messages are exchanged. A group cAKE scheme requires this property because the simulator cannot know a priori the group to which a simulated party belongs, and hence cannot know the "I am a member of group [...]" statement on which this party runs as a CKEM receiver R. However, once the functionality reveals e.g. that the simulated R is a member of the same group as the attacker, the simulator must complete the R simulation on such adaptively revealed statement.

(3) The third change is that we cannot disambiguate between proof/CKEM instances using *labels*, which were used to separate between honest and adversarial CKEM instances in e.g. [40]. This change stems from the fact that whereas in many contexts protocol instances can be tied to some public unique identifiers of participating parties, we cannot use such public identifiers in the context of covert authentication. We deal with this technical challenge by strengthening the strong simulation-soundness property (1) above even further, and requiring witness extractability from adversary \mathcal{A} which decrypts in interaction with a challenge S(x) instance, even if \mathcal{A} has access to (simulated) R(x') instances for any x' values, *including* x' = x, with the only constraint that no \mathcal{A} -R transcript equals the \mathcal{A} -S transcript. Note that the excluded case of such transcripts being equal corresponds to a *passive attack*, i.e. \mathcal{A} just transmitting messages between challenge oracles S and R, a case with which we deal separately.

We construct a covert CKEM, for any Sigma-protocol language, which satisfies this stronger covert CKEM notion, by using stronger building blocks compared to the (Sigma-protocol)-to-(Covert-CKEM) compiler of [39]. First, we rely on smooth projective hash functions (SPHF) with a property akin to PCA

¹² Covert CKEM was called ZKSend in [21]. Variants of (covert or non-covert) CKEM notion include Conditional OT [25], Witness Encryption [32], and Implicit ZK [9].

¹³ This requires a special-purpose commitment which is hiding only in the sense of one-wayness, and which allows linking a revocation token to a committed certificate.

(plaintext checking attack) security of encryption. Using Random Oracle hash in derivation of SPHF outputs it is easy to assure this property for standard SPHF's of interest. Secondly, we use covert trapdoor commitments, with commitment instances defined by a random oracle hash applied to CKEM statements, to enable postponed-statement simulation required by property (2) above. (Intuitively, trapdoor commitments allow the simulator to open a message sent on behalf of an honest party as a CKEM ciphertext corresponding to a group membership which the functionality reveals in response to a *subsequent* active attack against this party.)

We achieve low bandwidth of the fully instantiated group cAKE by instantiating the above with the Identity Escrow scheme implied by Pointcheval-Sanders (PS) group signatures [51]. The resulting IE certificates involve only two elements of a type-3 bilinear pairing curve [31], which can be covertly encoded using the Elligator Squared encoding of Tibouchi [59], with a hash onto group due to Wahby and Boneh [62]. The CKEM part (for the language of valid IE certificates) requires sending only 4 group elements (3 for R and 1 for S), and can be implemented over a standard curve, which can be covertly encoded using e.g. the Elligator-2 encoding of Bernstein et al. [11].

Restriction to static corruptions. We note that our group cAKE scheme realizes the UC group cAKE model only for the case of *static* corruptions, i.e. the adversary can compromise a certificate or reveal a corresponding revocation token only if this certificate has never been used by an honest party. This is because our group cAKE scheme has no *forward privacy or covertness*. In particular, all past sessions executed by a party on some certificate become identifiable, and hence lose covertness (but only covertness, and not security), if this certificate is compromised at any point in the future. This lack of forward privacy comes from the verifier-local revocation mechanism. Enabling forward privacy in the face of revocation, and doing so covertly, introduces new technical challenges. For example, we can use our CKEM for a covert proof that a committed certificate is (or is not) included on a most recent (positive or negative) accumulator (e.g. [50]) for a given group. However, it is not clear how two group members can *covertly* deal with a possible skew between the most recent accumulator values they assume. We leave solving such challenges to future work.

Related works. Von Ahn, Hopper, and Langford [61] introduced the notion of covert 2-party computation and achieved it by performing $O(\kappa)$ repetitions of Yao's garbled circuit evaluations. The underlying circuit was also extended by a hash function. This protocol guaranteed only secrecy against malicious participants and not output correctness. Chandran et al. [21] extended this to multiple parties while achieving correctness, but their protocol was also non-constant-round, and its efficiency was several orders of magnitude over known non-covert MPC protocols since each party covertly proves it followed a GMW MPC protocol by casting it as an instance of the Hamiltonian Cycle problem. Further, that proof internally used Yao's garbled circuits for checking correctness of committed values. Goyal and Jain [35] subsequently showed that non-constant-round protocols are necessary to achieve covert computation with black-box simulation against malicious adversaries, at least in the plain MPC model, i.e., without access to some trusted parameters. Hence, the former two constructions' inefficiencies are necessary without a trusted setup. Jarecki [39] showed a constant-round covert AKE with O(1) public key operations satisfying a game-based, group-based covert AKE definition with a trusted setup. This protocol has a somewhat large communication cost: three rounds and large bandwidth since it uses composite-order groups. Recently, Kumar and Nguyen [47] gave the first post-quantum covert group-based AKE with trusted setup by adopting Jarecki's construction [39] to a lattice-based construction (three rounds in the ROM). Kumar and Nguyen do not provide bandwidth estimates, but we expect them to be somewhat large compared to Jarecki's original construction since they rely on trapdoor lattices [33].

None of the aforementioned works are proven secure in the UC framework [19]. Cho, Dachman-Soled, and Jarecki [23] achieve UC security for covert MPC of two specific functionalities, namely string equality and set intersection. The work of Jarecki [40] achieves UC secure 2PC for any function, but its efficiency is constant-round and sends $O(\kappa |C|)$ symmetric ciphertexts and $O(n\kappa)$ group elements where C is a boolean circuit with n input bits for the function to be computed. Implementing covert group-based authenticated key exchange using such generic protocol would be exceedingly costly. An open question is if the covert group-based AKE of [39] is secure as-is in the UC model despite [39] using a weaker instantiation of a covert CKEM.

Organization. Section 2 provides preliminaries. Section 3 presents a universally composable (UC) model of group covert authenticated key exchange (group cAKE). Section 4 reviews the building blocks used in our construction, namely covert trapdoor commitments, SPHF's, and an Identity Escrow (IE). Section 5

uses the first two of these tools to construct a covert CKEM, a key modular component of our group cAKE. The group cAKE scheme itself is shown in Section 6. Appendices A and B contain the proofs of our two main security theorems, and Appendix C describes our proof of concept implementation.

2 Preliminaries

Let |p| be shorthand for the bit-length of a positive integer p. We reserve κ for the security parameter throughout the paper. We say a function $f(\kappa)$ on \mathbb{Z}^+ is negligible if it is asymptotically $\kappa^{-\omega(1)}$: for each polynomial q, there is an N, such that $f(\kappa) < 1/q(\kappa)$ for all $\kappa > N$. Furthermore, all probability distributions (\mathcal{X}) in this paper are from distribution families indexed by \mathbb{Z}^+ , or an infinite subset of \mathbb{Z}^+ , $\{\mathcal{X}_\kappa\}_{\kappa\in\mathbb{Z}^+}$. The uniform distribution on a finite set S is denoted as $\mathcal{U}(S)$, and we denote a random variable sampled from the uniform distribution over S as $x \leftarrow_{\mathbb{R}} S$, and $x \leftarrow_{\mathbb{R}} \mathcal{X}$ for a general distribution \mathcal{X} . We say two probability distributions, say \mathcal{X}_{κ} and \mathcal{Y}_{κ} , with the same domains are statistically close, denoted $\mathcal{X}_{\kappa} \approx_s \mathcal{Y}_{\kappa}$, if their normalized l_1 distance, $\mathcal{A}(\mathcal{X}_{\kappa}, \mathcal{Y}_{\kappa}) = \frac{1}{2} \sum_{z} |\mathcal{X}_{\kappa}(z) - \mathcal{Y}_{\kappa}(z)|$, is negligible as a function of κ . Two probability distributions, \mathcal{X} and \mathcal{Y} whose domains depend on some security parameter κ , are *computationally indistinguishable*, denoted $\mathcal{X} \approx_c \mathcal{Y}$, if for every probabilistic polynomial time algorithm \mathcal{A} , the following probability is negligible in κ

$$|\Pr\{\mathcal{A}(x) = 1 : x \leftarrow_{\mathsf{R}} \mathcal{X}\} - \Pr\{\mathcal{A}(y) = 1 : y \leftarrow_{\mathsf{R}} \mathcal{Y}\}|.$$

Our constructions rely on the Random Oracle Model (ROM) [6] where a hash function, $H : \{0,1\}^* \to D$, for some domain D, is modeled as a truly random function. Our constructions use several hash functions, H_1, H_2, \ldots, H_n , and an implementation must use domain separation in order to re-use a fixed hash function H, e.g., SHA-256, for multiple random oracles, e.g. $H_i(x) = H(\langle i \rangle || x)$, where $\langle i \rangle$ is some encoding of natural numbers. To obtain an RO hash H' for an arbitrary integer range $\mathbb{Z}_p = \{0, \ldots, p-1\}$ from an RO hash H with 2κ -bit outputs, one can set $H'(x) = (v \mod p)$ where $v = H(\langle 1 \rangle || x) || \ldots || H(\langle n \rangle || x)$ for any n s.t. $|p| \ge 2\kappa \cdot n + \kappa$.

2.1 Diffie-Hellman Problems

Here we review some background on Diffie-Hellman groups.

Definition 2.1. Let (\mathbb{G}, \cdot) be a cyclic group of order p with generator g. The computational Diffie-Hellman problem (CDH) on \mathbb{G} is given (g^a, g^b) for $a, b \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, compute g^{ab} . The computational Diffie-Hellman assumption on \mathbb{G} is that this problem is hard, i.e., not PPT adversary A can solve this problem with non-negligible probability for $\log_2 p \geq \kappa$.

Definition 2.2. Let (\mathbb{G}, \cdot) be a cyclic group of order p with generator g. The decisional Diffie-Hellman problem *(DDH)* on \mathbb{G} is to distinguish the the following two distributions:

$$\{(g, g^a, g^b, g^c) : a, b, c \leftarrow_{\scriptscriptstyle R} \mathbb{Z}_p\}$$

and

$$\{(g, g^a, g^b, g^{ab}) : a, b \leftarrow_{\scriptscriptstyle R} \mathbb{Z}_p\}$$

The decisional Diffie-Hellman assumption on \mathbb{G} is that this problem is hard, i.e., not PPT adversary A can solve this problem with non-negligible probability for $\log_2 p \ge \kappa$.

Definition 2.3. Let (\mathbb{G}, \cdot) be a cyclic group of order p with generator g. The gap computational Diffie-Hellman problem (gap-CDH) on \mathbb{G} is given (g^a, g^b) for $a, b \leftarrow_{\mathbb{R}} \mathbb{Z}_p$, compute g^{ab} given an oracle which decides DDH: $\mathcal{O}_{\text{DDH}}(g^a, g^b, g^c) = 1$ iff $c = ab \pmod{p}$. The computational gap Diffie-Hellman assumption on \mathbb{G} is that this problem is hard, i.e., not PPT adversary A can solve this problem with non-negligible probability for $\log_2 p \ge \kappa$.

Diffie-Hellman assumptions are commonly deployed on elliptic curve groups, such as Curve25519 [10].

2.2 Groups with Bilinear Maps

The key enabling mathematical structures throughout our paper are elliptic curves with type-3 bilinear pairings. First used by Joux [41] for cryptography, pairings allow for advanced cryptographic constructions due to their cryptographic hardness and bilinearity.

Definition 2.4. A bilinear pairing for groups of prime order p, \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

bilinearity: For all $g \in \mathbb{G}_1$, $\hat{g} \in \mathbb{G}_2$, $a, b \in \mathbb{Z}_p$, $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$. non-degeneracy: For all $g \neq 1_{\mathbb{G}_1}, \hat{g} \neq 1_{\mathbb{G}_2}, e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$. efficiency: $e(\cdot, \cdot)$ is efficiently computable.

A pairing is *Type 3* if there is no efficiently computable homomorphism between \mathbb{G}_1 and \mathbb{G}_2 [31]. We will use the following assumption, stated and proved in the generic group model [58] by Pointcheval and Sanders [51].

Assumption 1 ([51]) Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear pairing of type 3 with $g(\hat{g})$ a generator of \mathbb{G}_1 (\mathbb{G}_2), where they are generated according to a security parameter κ . In particular, their orders depend on κ . For $x, y \leftarrow \mathbb{Z}_p$ sampled uniformly and independently, set $\hat{X} = \hat{g}^x, \hat{Y} = \hat{g}^y$, and $Y = g^y$. We define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ to choose a random $h \in \mathbb{G}_1$ and output a pair $P = (h, h^{x+my})$. Then, the assumption is that no efficient adversary can generate a pair $(h_A, h_A^{x+m'y}) = P_A$ with non-negligible probability in κ given $(g, \hat{g}, Y, \hat{X}, \hat{Y})$ for any m' not previously queried to \mathcal{O} and $h_A \neq 1_{\mathbb{G}_1}$.

2.3 Special Honest-Verifier Zero-Knowledge Proofs: Σ -Protocols

A Σ -protocol is a special honest-verifier public-coin zero-knowledge proof of knowledge system [26] which forms a basis of many efficient ZKPK constructions. Here we define a stronger version of Σ -protocols than in [26] where (1) both the verifier and the simulator use the same deterministic function to recompute the prover's first message from the rest of the protocol's transcript, (2) the prover's last message is a deterministic function of prior messages, and (3) the simulator samples the prover's last message uniformly from a response space S_z .

Specifically, we define a Σ -protocol for \mathcal{R} as tuple ($\mathsf{P}_1, \mathsf{P}_2, \mathsf{VRec}, S_{ch}, S_z$), where P_1 is a randomized algorithm, $\mathsf{P}_2, \mathsf{VRec}$ are deterministic algorithms, and S_{ch}, S_z are sets, s.t. (1) for any $(x, w) \in \mathcal{R}$, algorithm $\mathsf{P}_1(x, w)$ samples prover's commitment a and randomness r, (2) verifier's challenge is generated by sampling ch uniformly in S_{ch} , (3) algorithm $\mathsf{P}_2(x, w, r, ch)$ outputs prover's response $z \in S_z$, and (4) the verifier accepts on statement x and transcript (a, ch, z) iff $\mathsf{VRec}(x, ch, z)$ outputs a. These algorithms must further satisfy:

- Completeness: For any $(x, w) \in \mathcal{R}$ and $ch \in S_{ch}$, if $(a, r) \leftarrow \mathsf{P}_1(x, w)$ and $z \leftarrow \mathsf{P}_2(x, w, r, ch)$ then $a \leftarrow \mathsf{VRec}(x, ch, z)$.
- Special Honest-Verifier Zero-Knowledge: For any $(x, w) \in \mathcal{R}$ and $ch \in S_{ch}$, transcript (a, z) generated by sampling $z \leftarrow_{\mathbb{R}} S_z$ and setting $a \leftarrow \mathsf{VRec}(x, ch, z)$, is distributed identically to an output of $(\mathsf{P}_1, \mathsf{P}_2)$ on (x, w, ch).
- Special Strong Soundness: Challenge space S_{ch} is super-polynomial (in security parameter), and an efficient extractor outputs w s.t. $(x, w) \in \mathcal{R}$ given any x and two accepting transcripts (a, ch, z) and (a, ch', z') for x s.t. $ch \neq ch'$.
- Response Uniqueness: For any x, ch, z, z', if (1) $ch \in S_{ch}$, (2) $z, z' \in S_z$, and (3) $\mathsf{VRec}(x, ch, z) = \mathsf{VRec}(x, ch, z')$, then z = z'.

As mentioned above, these properties are stronger than the original notion of Σ -protocol in [26], but they are satisfied by many well-known Σ -protocols for prime-order groups, including additive and multiplicative relations on discrete logarithms, see e.g. [26,18]. In this work we will utilize a Σ -protocol for relation $\mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$, see eq. (3), on elements of a bilinear map group, as shown in Section 4.3.

2.4 Covert Encodings and Random Beacons

We recall the covert encoding and random beacon notions used in steganography.

Definition 2.5. Functions (EC, DC) form a covert encoding of domain D if there is an l s.t. EC : $D \to \{0,1\}^l$, DC : $\{0,1\}^l \to D$ is an inverse of EC, and EC($\mathcal{U}(D)$) is statistically close to the uniform distribution on $\{0,1\}^l$. Function EC can be randomized but DC must be deterministic. In case EC is randomized we require EC($\mathcal{U}(D)$; r) to be statistically close to uniform when EC's randomness r is a uniform random bitstring of fixed length.

Definition 2.6. We call a finite set S uniformly encodable if it has a covert encoding. Further, a family of sets $S := \{S[\pi]\}_{\pi \in \mathcal{I}}$ indexed by some indexing set \mathcal{I} is uniformly encodable if $S[\pi]$ is uniformly encodable for each $\pi \in \mathcal{I}$.

Uniformly Encodable Domains. We use the following two uniformly encodable sets throughout the paper: (1) an integer range $[n] = \{0, ..., n-1\}$, and (2) points on an elliptic curve. For the former, if n is near a power of two then we can send an integer sampled in $\mathcal{U}([n])$ as is. Otherwise, for any t we can encode t-tuple $(a_i)_{i \in [t]}$ sampled from $\mathcal{U}([n]^t)$ as $\sum_{i=0}^{t-1} a_i \cdot n^i + r \cdot n^t$ for $r \leftarrow_{\mathbb{R}} [m]$ where $m = \lceil 2^{\log_2(n)+\kappa}/n \rceil$. (See e.g. Section 3.4 of [59] for a proof.) For uniform encodings of elliptic curve points we require two subcases: (2a) a curve in Montgomery form and (2b) a pairing friendly curve. In case (2a) we can use the Elligator-2 encoding [11], which takes a random point sampled from a subset S of group $\mathbb{G} = E(\mathbb{F}_p)$, where $|S|/|\mathbb{G}| \approx 1/2$, and injectively maps it to integer range [(p-1)/2]. This map is then composed with a uniform encoding of this integer range. In the random oracle model, if H is an RO hash onto \mathbb{G} , see e.g. [62], a simple way to encode point P sampled from the whole group, i.e. $P \leftarrow_{\mathbb{R}} \mathcal{U}(\mathbb{G})$ as opposed to $P \leftarrow_{\mathbb{R}} \mathcal{U}(S)$, is to sample $r \leftarrow_{\mathbb{R}} \{0,1\}^{\kappa}$ until Q = H(r) + P is in S, where G is a generator of \mathbb{G} , and output z = Elligator-2(Q)||r (see Lemma 2.1 below). In case (2b) we can use Tibouchi's Elligator Squared encoding [59], which represents a random curve point as a pair of random elements of base field \mathbb{F}_q . This randomized map is then composed with a uniform encoding of $[q]^2$, implemented as above. In summary, Elligator-2 admits a more narrow class of curves than Elligator Squared, but using the above methods, the former creates slightly shorter encodings than the latter, resp. $|p| + 2\kappa$ vs. $2|q| + \kappa$ bits.

Lemma 2.1. Let Elligator-2: $S \to [(p-1)/2]$ be the Elligator-2 map with $|S| \approx |\mathbb{G}|/2$. In the random oracle model, if H is an RO hash onto \mathbb{G} and $P \leftarrow_{\mathbb{R}} \mathcal{U}(\mathbb{G})$, then the outputs of the following algorithm are sampled from distribution $\mathcal{U}([(p-1)/2]) \times \mathcal{U}(\{0,1\}^{\kappa})$:

1. sample $r \leftarrow_{\mathbb{R}} \{0,1\}^{\kappa}$ until $Q = \mathsf{H}(r) + P$ is in S. 2. return z = Elligator-2(Q)||r.

Proof. We work backwards from $\mathcal{U}([(p-1)/2]) \times \mathcal{U}(\{0,1\}^{\kappa})$. First, take $(\alpha||r) \leftarrow \mathcal{U}([(p-1)/2]) \times \mathcal{U}(\{0,1\}^{\kappa})$ and invert Elligator-2 to get (Elligator- $2^{-1}(\alpha)||r) = (Q||r)$. This distribution is equal to $\mathcal{U}(S) \times \mathcal{U}(\{0,1\}^{\kappa})$. Next, note that for any r, the distribution of P := Q - H(r) is uniformly random over \mathbb{G} . This means that (P||r) = (Q - H(r)||r) is distributed as $\mathcal{U}(\mathbb{G}) \times \mathcal{U}(\{0,1\}^{\kappa})$.

Random Beacons. The term random beacon refers to a network node or party which broadcasts random bitstrings. Such randomness sources are used for covert communication and here we use it for covert authentication, and, more generally, covert computation. We use $B^{\$(\kappa)}$ where B is an interactive algorithm to denote a random beacon equivalent of B. Namely, if B has a fixed number of rounds and n_i is a polynomial s.t. for each *i*, the *i*-th round message of B has (at most) $n_i(\kappa)$ bits, then $B^{\$(\kappa)}$ is an interactive "algorithm" which performs no computation except for sending a random bitstring of length $n_i(\kappa)$ in round *i*.

3 Universally Composable Model for Group Covert AKE

As discussed in the introduction, we define group covert AKE (group cAKE) as a covert group Authenticated Key Exchange, i.e. a scheme which allows two parties certified by the same authority, a.k.a. a group manager, to covertly and securely establish a session key. Covert AKE must be as secure as standard AKE, i.e. an adversary who engages in sessions with honest parties and observes their outputs cannot break the security of any session except by using a compromised but non-revoked certificate. In addition, the protocol must be *covert* in the sense that an attacker who does not hold a valid and non-revoked certificate not only cannot authenticate to an honest party but also cannot distinguish interaction with that party from an interaction with a random beacon. If such protocol is implemented over a stegano-graphic channel [38] a party who does not have valid authentication tokens not only cannot use it to authenticate but also cannot detect if anyone else uses it to establish authenticated connections.

We define a group cAKE scheme as a tuple of algorithms (KG, CG, Auth) with the following input/output behavior:

- KG is a key generation algorithm, used by the group manager, s.t. $KG(1^{\kappa})$ generates the group public key, gpk, and a master secret key, msk.
- CG is a certificate generation algorithm, used by the group manager, s.t. CG(msk) generates a membership certificate *cert* with a revocation token *rt*.
- Auth is an interactive algorithm used by two group members to (covertly) run an authenticated key exchange. Each party runs Auth on local input $(gpk, cert, \mathsf{RTset})$, where RTset is a set of revocation tokens representing revoked parties. Each party outputs (K, rt), where $K \in \{0, 1\}^{\kappa} \cup \{\bot\}$ is a session key (or \bot if no key is established) and $rt \in \mathsf{RTset} \cup \{\bot\}$ is a detected revocation token in RTset , or \bot if Auth participant does not detect that a counterparty uses a certificate corresponding to a revocation token in RTset .

Our notion of AKE does not include *explicit entity authentication*, i.e., a party might output $K \neq \bot$ even though its counterparty is not a valid group member. However, since key K is secure, the parties can use standard key confirmation methods to explicitly authenticate a counterparty as a valid group member who computed the same session key. Moreover, Auth can remain covert even after adding key confirmation, e.g. if key confirmation messages are computed via PRF using key K. Note that in the definition above a real-world party P can output $K = \bot$, which violates the (simplified) covert mutual authentication model of Figure 1 in Section 1. However, w.l.o.g. P is free to run any upper-layer protocol Π that utilizes Auth output K by replacing $K = \bot$ with a random key, thus preserving its covertness if protocol Π is covert.

Universally Composable Group cAKE. We define security of group cAKE via a universally composable functionality $\mathcal{F}_{g\text{-}cAKE}$ shown in Figure 2, and we say that scheme $\Pi = (KG, CG, Auth)$ is a group cAKE if Π UC-realizes functionality $\mathcal{F}_{g\text{-}cAKE}$ in the standard sense of universal composability [19]. However, we adapt the UC framework [19] to the covert computation setting so that environment \mathcal{Z} can pass to party P executing an AKE protocol Auth a special input \bot , which causes party P to play a role of a random beacon. (The same convention was adopted by Chandran et al. [21] with regards to one-shot secure computation.) For simplicity of notation we assume that protocol Auth is symmetric, i.e., the two participants act symmetrically in the protocol, and that it has a fixed number of rounds. In this case, on input (NewSession, ssid, \bot) from \mathcal{Z} , this party's session indexed by identifier ssid is replaced by a random beacon, i.e., it will run Auth^{$\$(\kappa)$} instead of Auth, see Sec. 2

In Definition 3.1 we use the notation of [19], where $\mathbf{Ideal}_{\mathcal{F}_{g-cAKE},\mathcal{A}^*,\mathcal{Z}}(\kappa, z)$ stands for the output of environment \mathcal{Z} in the ideal-world execution defined by the ideal-world adversary (a.k.a. simulator) algorithm \mathcal{A}^* and functionality \mathcal{F}_{g-cAKE} , for security parameter κ and \mathcal{Z} 's auxiliary input z, and $\mathbf{Real}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ stands for \mathcal{Z} 's output in the real-world execution between a real-world adversary \mathcal{A} and honest parties acting according to scheme Π , extended as specified above in case party P receives \mathcal{Z} 's input (NewSession, ssid, \perp).

Definition 3.1. Protocol $\Pi = (KG, CG, Auth)$ realizes a UC Covert Authenticated Key Exchange if for any efficient adversary A there exists an efficient ideal-world adversary A^* such that for any efficient environment Z it holds that

 $\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{g-cAKE}},\mathcal{A}^*,\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}\approx_c\{\mathbf{Real}_{\varPi,\mathcal{A},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$

Group cAKE functionality. We explain how functionality $\mathcal{F}_{g\text{-}cAKE}$ operates and how it differs from a standard AKE functionality, e.g. [20,46]. Note that functionality $\mathcal{F}_{g\text{-}cAKE}$ in Figure 2 is much more complex than functionality $\mathcal{F}_{cAKE}[C_{Gwr}]$ in Figure 1 in Section 1. The first difference are environment commands Glnit and Certlnit, which are used to initialize groups and generate membership certificates, and commands CompCert and RevealRT, which model adversarial compromise of resp. certificates and \mathcal{F}_{g-cAKE} interacts with parties denoted P and GM, and adversary \mathcal{A}^* . Sets CompCert^{gid} and RevealRT^{gid} store resp. compromised certificates and revealed revocation tokens for each gid. Keys: Initialization and Attacks On (Glnit, gid) from GM: Save (gid, GM), reject future Glnit queries for the same gid, send (Glnit, GM, gid) to \mathcal{A}^* . On (CertInit, gid, cid) from P: If \exists no prior record (\cdot , gid, cid), save tuple (P, gid, cid). On (CompCert, P, gid, cid) from \mathcal{A}^* [\mathcal{A}^* must have environment permission for this action]: $\text{If } \exists \ \text{record} \ (\mathsf{P},\mathsf{gid},\mathsf{cid}) \ \text{and} \ \exists \ \text{no} \ \text{record} \ (\mathsf{P},\cdot,\mathsf{gid},\mathsf{cid},\cdot,\cdot) \ \text{add} \ \mathsf{cid} \ \text{to} \ \mathsf{CompCert}^{\mathsf{gid}} \ \text{and} \ \mathsf{Reveal}\mathsf{RT}^{\mathsf{gid}}.$ On (<u>RevealRT</u>, P, gid, cid) from \mathcal{A}^* [\mathcal{A}^* must have environment permission for this action]: If \exists record (P, gid, cid) and \exists no record (P, \cdot , gid, cid, \cdot , \cdot) add cid to RevealRT^{gid}. Authentication Sessions: Initialization, Connections, Attacks On (NewSession, ssid, \perp) from P: Save record $(\mathsf{P},\mathsf{ssid},\bot,\bot,\bot,\bot)$ marked random, send $(\mathsf{NewSession},\mathsf{P},\mathsf{ssid},\bot)$ to \mathcal{A}^* . On (NewSession, ssid, gid, cid, RTcids) from P: If $\mathsf{RTcids} \subseteq \mathsf{Reveal}\mathsf{RT}^{\mathsf{gid}}$ and \exists record (P, gid, cid) but \exists no prior record (P, ssid, $\cdot, \cdot, \cdot, \cdot)$: - if cid ∉ RevealRT^{gid}, send (NewSession, P, ssid, ⊥) to \mathcal{A}^* $- \text{ if } \mathsf{cid} \in \mathsf{RevealRT}^{\mathsf{gid}}, \, \mathrm{send} \, \, (\mathsf{NewSession}, \mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}) \, \, \mathrm{to} \, \, \mathcal{A}^*$ Save record (P, ssid, gid, cid, RTcids, \bot) marked fresh. On (Interfere, P, ssid) from \mathcal{A}^* : If \exists record $(\mathsf{P},\mathsf{ssid},\cdot,\cdot,\cdot,\bot)$ marked fresh, re-label it interfered. On (Connect, P, ssid, P', ssid') from \mathcal{A}^* : If \exists record rec = (P, ssid, gid, cid, \cdot, \bot) marked fresh and record (P', ssid', gid', cid', \cdot, K') marked either fresh or connected (P, ssid, cid) (any of gid', cid', K' can be equal to \perp) then: - if gid = gid' then re-label rec as connected(P', ssid', cid') - if $gid \neq gid'$ then re-label rec as interfered On (Impersonate, P, ssid, gid^{*}, cid^{*}) from \mathcal{A}^* : If $\exists rec = (\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \cdot, \cdot, \bot)$ marked fresh: $\begin{array}{l} - \mbox{ if } gid = gid^* \mbox{ and } cid^* \in CompCert^{gid} \mbox{ then re-label rec as compromised}(cid^*) \\ - \mbox{ if } gid = gid^* \mbox{ and } cid^* \not\in CompCert^{gid} \mbox{ then re-label rec as interfered}(cid^*) \end{array}$ - if $gid \neq gid^*$ then re-label rec as interfered Authentication Sessions: Key Establishment On (NewKey, P, ssid, K^*) from \mathcal{A}^* : If \exists session record rec = (P, ssid, gid, cid, RTcids, \perp) marked flag then: 1. if flag = random set $K \leftarrow \bot$ and $\operatorname{cid}_{\operatorname{CP}} \leftarrow \bot$ 2. if flag = compromised(cid') for cid' \notin RTcids, set $K \leftarrow K^*$ and cid_{CP} $\leftarrow \bot$ 3. if flag is either connected($\cdot, \cdot, \text{cid}'$) or compromised(cid') or interfered(cid'), for cid' $\in \mathsf{RT}$ cids, set $K \leftarrow \bot$ and $\operatorname{cid}_{\operatorname{CP}} \leftarrow \operatorname{cid}'$ 4. if flag = connected(P', ssid', cid') for cid' \notin RTcids, and \exists rec' = (P', ssid', gid, cid', \cdot, K') s.t. $K' \neq \bot$ and rec' terminated as connected(P, ssid, cid), set $K \leftarrow K'$ and $cid_{CP} \leftarrow \bot$ 5. in any other case set $K \leftarrow_{\mathbb{R}} \{0,1\}^{\kappa}$ and $\mathsf{cid}_{\mathsf{CP}} \leftarrow \bot$ Modify rec as $(\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids}, K)$ and output $(\mathsf{NewKey}, \mathsf{ssid}, K, \mathsf{cid}_{\mathsf{CP}})$ to P .

Fig. 2. \mathcal{F}_{g-cAKE} : Group cAKE functionality, static corruptions enforced by boxed text

revocation tokens (which are not assumed public by default). Command NewSession models party P engaging in group cAKE on input $x = (gpk, cert, \mathsf{RTset})$, exactly as $\mathcal{F}_{cAKE}[\mathsf{C}_{Gwr}]$ of Figure 1, except that in $\mathcal{F}_{g\text{-}cAKE}$ these real-world inputs are replaced by ideal-world identifiers, resp. gid, cid, RTcids. One aspect of functionality $\mathcal{F}_{g\text{-}cAKE}$ is that there can be many number of such sessions present, and the adversary can "connect" any pair of such sessions, by passing their messages. Secondly, the adversary can actively attack any session using some compromised group certificate, and functionality $\mathcal{F}_{g\text{-}cAKE}$ carefully delineates the effect of such attack based on whether the group assumed by the attacker matched the one used by the attacker party, and if so then whether the certificate used by the attacker was revoked by the attacked party.

Secure initialization and trusted group manager. A crucial difference between \mathcal{F}_{g-CAKE} and standard AKE is that in the latter each party can function on its own, creating its (private, public) key pair, e.g. as in [37], maybe accessing a global certificate functionality, e.g. as in [20]. By contrast, the Covert AKE model \mathcal{F}_{g-CAKE} must explicitly include a group manager party, denoted GM, initialized via query (Glnit, gid) which models generation of a group public key indexed by a unique identifier gid. Consequently, the \mathcal{F}_{g-CAKE} model assumes a trusted party, secure channels at initialization, and secure distribution of revocation tokens. We explain each of these assumptions in turn. Note that identifier gid in command (Glnit, gid) is associated with that group instance by each party P, which can be realized if GM has a reliable authenticated connection to each party, which allows authenticated broadcast of gpk. GM is assumed trusted because the model does not allow a compromise of GM or the master secret msk held by GM. Furthermore, when \mathcal{Z} 's command (Certlnit, gid, cid) to party P, prompting it to generate a membership certificate with identifier cid (assumed unique within group gid), we assume that only P can later use it to authenticate. Looking ahead, we will implement Certlnit relying on a secure channel between P and GM. Party GM will generate the certificate identified by cid, it will send it to P on the secure channel, and GM will be trusted not to use the certificate itself.

The above assumptions pertain to initialization procedures, but the on-line authentication will rely on the secure P-to-GM channels in one more aspect, namely for secure delivery of revocation tokens. The environment tells P to run the authentication protocol via query (NewSession, ssid, gid, cid, RTcids), which models P starting an AKE session using its certificate identified by cid within group gid, where RTcids is a set of identifiers of revoked certificates which P will use on this session. Crucially, at this step an implementation must allow P to translate this set of certificate identifiers RTcids into a set RTset of actual revocation tokens corresponding to these certificates. This can be realized e.g. if the trusted party GM stores the revocation tokens for all certificates it generates and that the P-GM channel allows for secure and authenticated transmission of the revocation tokens from GM to P whenever the environment requests it by including them in set RTcids input to P in some NewSession query. Note that the environment can set RTcids in an arbitrary way, which models e.g. parties that do not receive the revocation tokens of all compromised parties.¹⁴

Static compromise model. Adversary can compromise any certificate, using command (CompCert, gid, P, cid), and it can reveal the revocation information corresponding to any certificate, using command $(\mathsf{RevealRT},\mathsf{gid},\mathsf{P},\mathsf{cid}). \ \text{The first command adds cid to the set } \mathsf{CompCert}^{\mathsf{gid}} \ \text{of compromised certificate}$ identifiers in group gid, and both commands add cid to the set RevealRT^{gid} of certificate identifiers whose revocation tokens are revealed to the adversary. A compromised certificate cid allows the adversary to actively authenticate to other parties using interface Impersonate, whereas a revealed revocation token implies that party P which uses it to authenticate can be identified by the adversary, and hence no longer covert (see the second clause in NewSession interface). Finally, we allow only for *static* corruptions, which is implied by marked text fragments in Fig. 2, which impose that an adversary can compromise a certificate and/or reveal a revocation token only if this certificate was never used by an honest party. This is because the group cAKE scheme we show in this work has no forward privacy, i.e., all past sessions executed by a party on some certificate become identifiable, and hence lose covertness, if this certificate is compromised at any point in the future. Because it appears difficult to capture a notion of "revocable covertness", i.e., that protocol instances remain covert until a certificate they use is revealed, we forego on trying to capture such property and limit the model by effectively requiring that the adversary corrupts all certificates and reveals all revocation tokens at the beginning of the interaction.

¹⁴ To see an example of how real-world parties can use scheme $\Pi = (KG, CG, Auth)$ to implement the environment's queries to \mathcal{F}_{g-cAKE} , please see Figure 5 in Section 6.

Note on the environment. An environment plays a role of an arbitrary application utilizing the group cAKE scheme. The role of group cAKE is to make real AKE sessions indistinguishable from random beacons, but the two send different outputs to the environment: the former outputs keys, the latter do not. If the environment leaks that output to the adversary then the benefit of covertness will disappear. However, this is so in the real-world: If an adversary can tell that two nodes use the established key to communicate with each other, they will identify these parties on the application level and the covert property of the AKE level was "for naught", at least in that instance. However, if the upper-layer communication stays successfully hidden in some steganographic channel, then the adversary continues being unable to detect these parties. The versatility of a universally composable definition is that it implies the maximum protection whatever the strength of the upper-layer application: If the upper-layer allows some sessions to be detected (or even leaks the keys they use), this information does not help to detect other sessions, and it does not help distinguish anything from the cryptographic session-establishment protocol instances. The same goes for the revocation information the AKE sessions take as inputs: If the upper-layer detects compromised certificates and delivers the revocation information to all remaining players, the adversary will fail to authenticate to other group members and it will fail to distinguish their session instances from random beacons. If the revocation information does not propagate to some group member, the adversary can detect that party using a compromised certificate, but this inevitable outcome will not help the attacker on any other sessions.

3.1 Detailed Walk Through \mathcal{F}_{g-cAKE} Session Interfaces

Below we provide a detailed walk through session establishment, attacks, and termination interfaces of functionality \mathcal{F}_{g-cAKE} in Figure 2.

AKE session establishment and attacks:. Party P starts an AKE session by environment's command (NewSession, ssid, gid, cid, RTcids), where ssid is a locally unique session identifier (which w.l.o.g. can be implemented by a counter), gid identifies the group instance, cid is an identifier of a membership certificate P will use on this session, and RTcids contains of identifiers of revoked certificates. (In our implementation these identifiers will be translated into revocation tokens reliably delivered from GM.) For each session which uses a non-compromised/revoked certificate, i.e., if cid \notin RevealRT^{gid}, \mathcal{F}_{g-cAKE} reveals only (P, ssid) to \mathcal{A}^* . Note that entity P and session counter ssid are just handles on a unique protocol instance, but this information does not include either the instance gid of the group public key or the identifier cid of the membership certificate. The same origin information (P, ssid) is leaked to \mathcal{A}^* if the session is a random beacon, which happens if \mathcal{Z} starts P via command (NewSession, ssid, \perp), hence \mathcal{A}^* cannot tell an AKE session from a random beacon and has to simulate either one in the same way. However, if cid \in RevealRT^{gid}, i.e., P uses a certificate whose revocation token leaked to the adversary, \mathcal{F}_{g-cAKE} reveals (gid, cid) to \mathcal{A}^* , i.e., the protocol instance is non-covert and can be linked to the group gid and a particular certificate cid. The functionality also marks random-beacon sessions random and real sessions fresh, but does not leak it to \mathcal{A}^* .

After session (P, ssid) starts, the adversary can do 3 things to it: First, is that it can perform some active attack but without successfully following the protocol on a valid certificate. All such attempts should be equivalent to a denial-of-service attack, and they are modeled by query (Interfere, P, ssid) interface which marks such session interfered. The consequence of that is that when the session terminates it will output a random key unknown to anyone else (see below on AKE session termination). The second option is that the real-world adversary routes all messages between session (P, ssid) and some other session $(\mathsf{P}',\mathsf{ssid}')$, thus letting them communicate, a.k.a. "connect", at least until the last P' -to- P message. This is modeled by query (Connect, P, ssid, P', ssid'). Note that the adversary can do that to both the real sessions and the random beacon sessions. (Indeed, unless a real session runs on credentials whose revocation tokens were leaked, the adversary cannot tell the difference between the two.) In response \mathcal{F}_{g-cAKE} will mark the (P, ssid) session as connected to (P', ssid') but it will do so only if (1) both session are non-random and they assume the same group instance gid = gid', (2) session (P, ssid) is fresh, i.e., the adversary hasn't interrupted it before, and (3) session (P', ssid') is also fresh or it was already connected to (P, ssid). If either condition is not met this attempted connection will result in session (P, ssid) being marked interfered (so it outputs an independent random key at termination), but note that \mathcal{A}^* does not learn which is the case. The last option is that the adversary interacts with session (P, ssid) using a compromised certificate with identifier cid^{*} for group gid^{*}, which is modeled by command (Impersonate, P, ssid, gid^{*}, cid^{*}). The functionality checks that cid^* is indeed a compromised certificate generated within group gid^* . In that case, if the attacked session (P, ssid) runs on the same group instance $gid = gid^*$ then \mathcal{F}_{g-cAKE} marks session (P, ssid) as compromised using certificate cid^* , and otherwise it treats it just like an interfered session. Note that the attacker again does not learn which is the case, i.e. it does not learn if $gid = gid^*$.

AKE session termination:. Finally, if the real-world adversary sends to a session all the messages it needs to terminate, the ideal-world adversary sends (NewKey, P, ssid, K^*) to \mathcal{F}_{g-cAKE} , where K^* is the extracted key which the honest session (P, ssid) would compute *if* it was successfully attacked. There are 6 cases for such sessions:

(1) if the session is random, i.e., it was a random beacon, then its outputs are \perp no matter what;

(2,3) if the session is marked compromised(cid'), i.e., it was compromised by an active attack using matching gid and a compromised certificate cid', then (3) if cid' is in the list RTcids of revoked certificates P uses then (P, ssid) will detect this and reject by outputting $K = \bot$ and the detected certificate cid' as the *counterparty certificate identifier* cid_{CP}, but (2) if cid' is not on the list, i.e., the adversary used a certificate which P does not have on its revocation list, then this authentication succeeds and P outputs the adversarial key $K = K^*$ (and cid_{CP} = \bot since P did not detect its counterparty's certificate);

(3,4) if the session is marked connected(P', ssid', cid'), i.e., it was passively connected to another session running on the same gid, then (3) if that session used certificate cid' on P's revocation list RTcids then P rejects. However, if cid' is not on that list then (4) if (P', ssid') was the first to terminate, and it established a key $K' \neq \perp$, then \mathcal{F}_{g-CAKE} copies this key and gives it to P as well, i.e., sets $K \leftarrow K'$: This is the case of two AKE sessions which run on mutually non-revoked valid certificates for the same group gid and thus successfully establish a shared secure key;

(5) if (P, ssid) is connected to (P', ssid') but is the first to terminate, or if (P, ssid) is "left alone", or its session was interrupted, this party outputs a secure key.

Note that the only case when \mathcal{A}^* learns the key is case (2), an unavoidable on-line attack using a nonrevoked certificate. In all other cases a session either rejects (1,3) or outputs a secure key (5) which can be computed by two parties (4) if they run on compatible inputs and all their messages are delivered without interference.

4 Building Blocks: Commitment, SPHF, Identity Escrow

Our group cAKE construction consists of (1) each party sending out a blinded covert Identity Escrow (IE) certificate, and (2) each party verifying the counterparty's value using a covert Conditional Key Encapsulation Mechanism (CKEM). (This group cAKE construction is shown in Figure 6 in Section 6.) The covert CKEM construction in turn uses a covert Trapdoor Commitment and a covert Smooth Projective Hash Function (SPHF) which must be secure against a Plaintext Checking Attack (PCA). In this section we define and show efficient instantiations for each of the three above building blocks, i.e. covert Trapdoor Commitments, in Subsection 4.1, PCA-secure covert SPHF, in Subsection 4.2, and covert IE, in Subsection 4.3. (The construction of covert CKEM using trapdoor commitments and PCA-secure SPHF is shown in Section 5.) To fit bandwidth restrictions of steganographic channels we instantiate all tools with bandwidth-efficient schemes, using standard prime-order elliptic curve group for the Trapdoor Commitment and SPHF, and type-3 curves with bilinear pairings for IE.

4.1 Covert Trapdoor Commitment

For the reasons we explain below, we modify the standard notion of a Trapdoor Commitment [30] by splitting the commitment parameter generation into two phases. First algorithm GPG on input the security parameter κ samples global commitment parameters π , and then algorithm PG on input π samples instance-specific parameters $\overline{\pi}$. The commitment and decommitment algorithms then use pair $(\pi, \overline{\pi})$ as inputs. The trapdoor parameter generation TPG runs on the global parameters π output by GPG,

but it generates instance parameters $\overline{\pi}$ with the trapdoor tk. Then, the trapdoor commitment algorithm TCom on input π generates commitment c with a trapdoor td, and the trapdoor decommitment algorithm TDecom on input $(\pi, \overline{\pi}, c, tk, td, m)$ generates decommitment d. Crucially, the trapdoor commitment TCom takes only global parameters as inputs, which allows a simulator to create trapdoor commitments independently from the instance parameters $\overline{\pi}$.

Definition 4.1. Algorithm tuple (GPG, PG, Com, Decom) forms a trapdoor commitment scheme if there exists algorithms (TPG, TCom, TDecom) s.t.:

- $\mathsf{GPG}(1^{\kappa})$ samples global parameters π and defines message space \mathcal{M}
- $\mathsf{PG}(\pi)$ samples instance parameters $\overline{\pi}$
- $\mathsf{Com}(\pi, \overline{\pi}, m)$ outputs commitment c and decommitment d
- $\mathsf{Decom}(\pi, \overline{\pi}, c, m, d)$ outputs 1 or 0
- $\mathsf{TPG}(\pi)$ outputs instance parameters $\overline{\pi}$ with trapdoor tk
- $\mathsf{TCom}(\pi)$ outputs commitment c with trapdoor td
- $\mathsf{TDecom}(\pi, \overline{\pi}, c, tk, td, m)$ outputs decommintment d

The correctness requirement is that if $\pi \leftarrow \mathsf{GPG}(1^{\kappa}), \ \overline{\pi} \leftarrow \mathsf{PG}(\pi), \ and \ (c,d) \leftarrow \mathsf{Com}(\pi,\overline{\pi},m) \ then \mathsf{Decom}(\pi,\overline{\pi},c,m,d) = 1.$

Definition 4.2. We say that a trapdoor commitment scheme forms a covert perfectly-binding trapdoor commitment *if it satisfies the following:*

- 1. Trapdoored and non-trapdoored distributions indistinguishability: For any m tuples $(\pi, \overline{\pi}, c, d)$ generated by the following two processes are computationally indistinguishable: sample $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$ and fix any $m \in \mathcal{M}$,
 - $\begin{array}{rcl} P_0 & : & \overline{\pi} \leftarrow \mathsf{PG}(\pi), (c,d) \leftarrow \mathsf{Com}(\pi,\overline{\pi},m) \\ P_1 & : & (\overline{\pi},tk) \leftarrow \mathsf{TPG}(\pi), (c,td) \leftarrow \mathsf{TCom}(\pi), \\ & & d \leftarrow \mathsf{TDecom}(\pi,\overline{\pi},c,tk,td,m) \end{array}$
- Perfect binding: If π ← GPG(1^κ) and π ← PG(π), then for any c, m, m', d, d' it holds except for negligible probability over the coins of GPG and PG, that if Decom(π, π, c, m, d) = Decom(π, π, c, m', d') = 1 then m = m'.
- 3. Covertness: There is a uniformly encodable set family S s.t. for any m, tuples $(\pi, \overline{\pi}, c)$ and $(\pi, \overline{\pi}, c')$ are computationally indistinguishable for $\pi \leftarrow \mathsf{GPG}(1^{\kappa}), \ \overline{\pi} \leftarrow \mathsf{PG}(\pi), \ c \leftarrow \mathsf{Com}(\pi, \overline{\pi}, m), \ c' \leftarrow_{\mathbb{R}} \mathcal{U}(S[\pi]).$

Discussion. The first property is specialized for scenarios where each commitment instance $\overline{\pi}$ is used only for a single commitment. This restriction is not necessary for the implementation shown below, but we use it for simplicity because it suffices in our CKEM application. Note that *perfect binding* property holds on all non-trapdoored commitment instance parameters $\overline{\pi}$, and it is unaffected by the equivocability of commitments pertaining to any trapdoored commitment instances $\overline{\pi'}$. Observe also that the *covertness* property implies the standard *computational hiding* property of the commitment. Finally, we note that the above properties do not imply non-malleability, and we defer to Section 5 for the intuition why that suffices in the CKEM application.

Random Oracle Applications. In the Random Oracle Model (ROM) it can be convenient to replace the instance generator algorithm PG with a random oracle, but for that we need an additional property:

Definition 4.3. We say that a trapdoor commitment scheme has RO-compatible instance parameters if each π output by $\mathsf{GPG}(1^{\kappa})$ defines set $\mathcal{C}[\pi]$ s.t. (1) distribution $\{\overline{\pi}\}_{\overline{\pi} \leftarrow \mathsf{PG}(\pi)}$ is computationally indistinguishable from uniform in $\mathcal{C}[\pi]$, and (2) there exists an RO-indifferentiable hash function $\mathsf{H}: \{0,1\}^* \to \mathcal{C}[\pi].$

The above property allows an application to set instance parameters as $\overline{\pi} := H(IbI)$, where string IbI can be thought of as a *label* of that commitment instance. If a label can be uniquely assigned to a committing party then for all labels corresponding to adversarial instances the simulator can set H(IbI) by sampling $\mathsf{PG}(\pi)$, which makes all these instances perfectly binding, while for all labels corresponding to honest parties the simulator can set H(IbI) by sampling $TPG(\pi)$, which makes all these instances equivocable.

In the CKEM application, Section 5, the label |b| is a statement x used in a given CKEM instance. In this way the simulator can "cheat" in the CKEM's on statements of the simulated parties without affecting the soundness of the CKEM's executed by the adversarial parties.¹⁵ The same CKEM application also motivates why it is useful for the trapdoor commitment TCom to be independent of a commitment instance parameter $\overline{\pi}$. Namely, this enables the "statement-postponed zero-knowledge" property in the CKEM application, where the simulator at first does not know the statement x used by the CKEM sender on the onset of simulation, but it can use $\mathsf{TCom}(\pi)$ to create an equivocable commitment, which it can then open to an arbitrary message for any parameter $\overline{\pi} = H(x)$ generated in the trapdoored way.

Covert Trapdoor Commitment Instantiation under DDH in ROM. We instantiate the covert trapdoor commitment using a modified Pedersen commitment which is equivocable on trapdoored instances and perfectly binding on the non-trapdoored instances. The instantiation relies on a group of prime order which is uniformly encodable and on which the DDH assumption holds.¹⁶

Construction 4.1 Define a 'Double Pedersen' trapdoor commitment scheme:

- $\mathsf{GPG}(1^{\kappa})$ fixes group \mathbb{G} of prime order q, picks $g_1, g_2 \leftarrow_{\scriptscriptstyle R} \mathbb{G} \setminus \{1\}$, and sets $\pi = (\mathbb{G}, q, g_1, g_2)$ and $\mathcal{M} = \mathbb{Z}_a$
- $\mathsf{PG}(\pi)$ outputs $\overline{\pi} = (h_1, h_2)$ for $h_1, h_2 \leftarrow_{\scriptscriptstyle R} \mathbb{G}$
- $\begin{array}{l} -\operatorname{\mathsf{Com}}(\pi,\overline{\pi},m) \text{ outputs } (c,d) \text{ for } d \leftarrow_{\scriptscriptstyle R} \mathbb{Z}_q \text{ and } c \leftarrow (g_1^d \cdot h_1^m, g_2^d \cdot h_2^m) \\ -\operatorname{\mathsf{Decom}}(\pi,\overline{\pi},c,m,d) \text{ outputs } 1 \text{ iff } m \in \mathbb{Z}_q \text{ and } c = (g_1^d \cdot h_1^m, g_2^d \cdot h_2^m) \end{array}$
- $\mathsf{TPG}(\pi)$ picks $tk \leftarrow_{\mathbb{R}} \mathbb{Z}_q$ and outputs $\overline{\pi} = (h_2, h_2) = (g_1^{tk}, g_2^{tk})$ and tk- $\mathsf{TCom}(\pi)$ picks $td \leftarrow_{\mathbb{R}} \mathbb{Z}_q$ and outputs $c = (g_1^{td}, g_2^{td})$ and td
- $\mathsf{TDecom}(\pi, \overline{\pi}, c, tk, td, m)$ outputs $d = td tk \cdot m \mod q$

Theorem 4.1. The 'Double Pedersen' trapdoor commitment construction 4.1 is covert perfectly-binding with RO-compatible instance parameters, assuming that the DDH assumption holds in group \mathbb{G} , that \mathbb{G} is uniformly encodable, and that there is an RO-indifferentiable hash onto \mathbb{G} .

Proof. We argue the four properties separately. To argue trapdoored and non-trapdoored indistinguishability of tuple $(\pi, \overline{\pi}, c, d)$, denote the non-trapdoored view as V_0 and the trapdoored view as V_1 . Consider a modified trap doored view V_1' where $\overline{\pi}$ is generated by TPG but c and d are generated by Com instead of by TCom and TDecom. Note that $V'_1 \equiv V_1$, because if $(h_1, h_2) = (g_1^{tk}, g_2^{tk})$ then $c = (g_1^d \cdot h_1^m, g_2^d \cdot h_2^m) = (g_1^{td}, g_2^{td})$ for $td = d + tk \cdot m$, which means that a standard commitment looks exactly like a trapdoored commitment, moreover d is the same value $d = td - tk \cdot m$ which is revealed in trapdoored decommitment. Note that V_0 and V'_1 differ only in generation of strings $\overline{\pi} = (h_1, h_2)$, which are generated as random pairs of group elements in V_0 and as (g_1^{tk}, g_2^{tk}) for $tk \leftarrow_{\mathbb{R}} \mathbb{Z}_q$ in V'_1 . Since (g_1, g_2) are random group elements, indistinguishability of V_0 and V'_1 follows from DDH assumption on the group.

Perfect binding holds because if (h_1, h_2) is generated by PG than if $(\alpha_1, \alpha_2) = (\mathsf{dlog}(g_1, h_1), \mathsf{dlog}(g_2, h_2))$ then $\alpha_1 \neq \alpha_2$ except for 1/q probability, in which case for every $c \in \mathbb{G}^2$ there exists a unique $(d, m) \in \mathbb{Z}_q^2$ s.t. $c = (g_1^d h_1^m, g_2^d h_2^m)$, namely (d, m) s.t. $(d + \alpha_1 m, d + \alpha_2 m) = (\gamma_1, \gamma_2)$ where $c = (g_1^{\gamma_1}, g_2^{\gamma_2})$.

To argue *covertness*, assume the uniformly encodable set \mathbb{G}^2 . (Note that if random \mathbb{G} elements are encodable as random bitstrings then so are random pairs of G elements.) Note that $c = (c_1, c_2)$ output by Com is generated s.t. $(c_1/h_1^m, c_2/h_2^m) = (g_1^d, g_2^d)$ for random $d \leftarrow_{\mathbb{R}} \mathbb{Z}_q$, while for c which is uniform in \mathbb{G}^2 pair $(c_1/h_1^m, c_2/h_2^m)$ is uniform in \mathbb{G}^2 . Since g_1 and g_2 are random elements of \mathbb{G} , indistinguishability of these two distributions follows by reduction to DDH.

It follows that the scheme has *RO-compatible instance parameters*, because algorithm $\mathsf{PG}(\pi)$ samples parameters $\overline{\pi}$ at random in space $\mathcal{C} = \mathbb{G} \times \mathbb{G}$, and if H is an RO-indifferentiable hash onto \mathbb{G} then H' defined e.g. as $\mathsf{H}'(x) = (\mathsf{H}(0|x), \mathsf{H}(1|x))$ is an RO-indifferentiable hash onto $\mathbb{G} \times \mathbb{G}$.

¹⁵ Except if an adversarial party copies a statement of the honest party, in which case CKEM security comes from the PCA security of SPHF, see Section 4.2.

¹⁶ Looking ahead, for OW-PCA security of SPHF for a language associated to this commitment, see Section 4.2, we will also need the GapDH assumption on the group.

4.2 Covert SPHF with PCA-security

A smooth projective hash function (SPHF) for an NP language \mathcal{L} , introduced by Cramer and Shoup [24], allows two parties to compute a hash on a statement $x \in \mathcal{L}$ where one party computes the hash using a random hash key hk and the statement x, and the other can recompute the same hash using a *projection* key hp corresponding to hk and a witness w for $x \in \mathcal{L}$. The smoothness property is that if $x \notin \mathcal{L}$ then the hash value computed using key hk on x is statistically independent of the projection key hp. In other words, revealing the projection key hp allows the party that holds witness w for $x \in \mathcal{L}$ to compute the hash value, but it hides this value information-theoretically if $x \notin \mathcal{L}$. In this work we require two additional properties of SPHF, namely covertness and One-Wayness under Plaintext Checking Attack (OW-PCA) security, which we define below.

Definition 4.4. A covert smooth projective hash function (covert SPHF) for NP language \mathcal{L} parameterized by π , is a tuple of PPT algorithms (HKG, Hash, PHash) and set family \mathcal{H} indexed by π , where HKG(π) outputs (hk, hp), and PHash(x, w, hp) and Hash(x, hk) both compute a hash value v s.t. $v \in \mathcal{H}[\pi]$. Furthermore, this tuple must satisfy the following properties:

- Correctness: For any (π, x, w) s.t. $x \in \mathcal{L}[\pi]$ and w is a witness for x, if $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$ then $\mathsf{Hash}(x, hk) = \mathsf{PHash}(x, w, hp)$.
- Smoothness: For any π and $x \notin \mathcal{L}[\pi]$, hash $\mathsf{Hash}(x, hk)$ is statistically close to uniform over $\mathcal{H}[\pi]$ even given hp, i.e. tuples (hp, v) and (hp, v') are statistically close for $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$, $v \leftarrow \mathsf{Hash}(x, hk)$, and $v' \leftarrow_{\mathbb{R}} \mathcal{U}(\mathcal{H}[\pi])$. Moreover, space $\mathcal{H}[\pi]$ must be super-polynomial in the length of π .
- Covertness: There is a uniformly encodable set S s.t. for any π , distribution $\{hp\}_{(hk,hp) \leftarrow_R \mathsf{HKG}(\pi)}$ is statistically close to uniform over $S[\pi]$.

One-Wayness under Plaintext-Checking Attack (OW-PCA) for SPHF. We define OW-PCA security notion for SPHF in analogy with OW-PCA security of Key Encapsulation Mechanism (KEM). OW-PCA security of KEM [34,53] asks that for a random KEM public key pk and ciphertext c, an efficient attacker cannot, except for negligible probability, output the key k encrypted in c even given access to a Plaintext-Checking (PCA) oracle, which holds the corresponding secret key sk and for any (ciphertext,key) query (c', k') outputs 1 if k' = Dec(sk, c') and 0 otherwise. An SPHF can implement a KEM if \mathcal{L} is hard on average, i.e. if on random $x \in \mathcal{L}$ it is hard to compute the corresponding witness w, because statement x, witness w, projection key hp, and hash value v could play the KEM roles of respectively pk, sk, c, and k. We define the OW-PCA property of SPHF as requiring that such KEM scheme is OW-PCA secure, i.e. that for a random (statement, witness) pair (x, w) in \mathcal{L} and random HKG (π) outputs (hk, hp), an efficient attacker cannot output v = Hash(x, hk) even given access to a PCA oracle, which holds the witness w and for any query (hp', v') outputs 1 if v' = PHash(x, w, hp') and 0 otherwise.

Following the above parallel to the OW-PCA property of KEM, statement x, which acts like a public key, should be randomly sampled by the challenger. However, in the CKEM applications of Section 5, we need OW-PCA SPHF for statements chosen from a "mixed" distribution, where part the statement is arbitrarily chosen by the adversary and only part is randomly sampled by the challenger. Specifically, we will consider language $\mathcal{L}^{\mathsf{Com}}$ of valid commitments in a covert perfectly-binding trapdoor commitment scheme, see Definition 4.2, parameterized by global commitment parameters π :

$$\mathcal{L}^{\mathsf{Com}}[\pi] = \{ (\overline{\pi}, m, c) \mid \exists \ d \ \text{s.t.} \ \mathsf{Decom}(\pi, \overline{\pi}, c, m, d) = 1 \}$$
(1)

Further, we will need OW-PCA security to hold for statements $x = (\overline{\pi}, m, c)$ where components $(\overline{\pi}, m)$ are chosen by the adversary on input π while component c together with witness d is chosen at random by the OW-PCA challenger.

In general, let \mathcal{L} be parameterized by strings π sampled by alg. $\mathsf{PG}_{\mathsf{sphf}}(1^{\kappa})$, let $\mathcal{L}_{\mathsf{pre}}[\pi]$ be a language of fixed-length prefixes of elements in $\mathcal{L}[\pi]$, and for any π and $x_L \in \mathcal{L}_{\mathsf{pre}}[\pi]$, let

$$\mathcal{R}_{\mathcal{L}}[\pi, x_L] = \{ (x_R, w) \mid \text{s.t.} \ (x_L, x_R) \in \mathcal{L}[\pi] \text{ and } w \text{ is its witness} \}.$$

Notably from in Equation 1, $x_L = (\overline{\pi}, m)$, $x_R = c$, and the witness w is the decommitment d. We define OW-PCA of SPHF for \mathcal{L} as follows:

 $\textbf{Definition 4.5. } \textit{SPHF for language \mathcal{L} with parameter generation algorithm $\mathsf{PG}_{\mathsf{sphf}}$ and prefix language \mathcal{L} with parameter generation algorithm $\mathsf{PG}_{\mathsf{sphf}}$ and prefix language \mathcal{L} with parameter generation algorithm $\mathsf{PG}_{\mathsf{sphf}}$ and prefix language $\mathcal{L}_{\mathsf{sphf}}$ and prefix language $\mathsf{PG}_{\mathsf{sphf}}$ and prefix language $\mathcal{L}_{\mathsf{sphf}}$ language $\mathcal{L}_{\mathsf{sphf}}$$ \mathcal{L}_{pre} is One-Way under Plaintext Checking Attack (OW-PCA) if for any efficient A the following probability is negligible:

 $Pr\left[v = \mathsf{Hash}(x, hk) \mid v \leftarrow \mathcal{A}^{\mathsf{PCA}(w, \cdot)}(\pi, x, hp, st)\right]$

where $\pi \leftarrow \mathsf{PG}_{\mathsf{sphf}}(1^{\kappa}), \ (x_L, st) \leftarrow \mathcal{A}(\pi) \ s.t. \ x_L \in \mathcal{L}_{\mathsf{pre}}[\pi], \ (x_R, w) \leftarrow_{\scriptscriptstyle R} \mathcal{R}_{\mathcal{L}}[\pi, x_L], \ x \leftarrow \ (x_L, x_R),$ $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$, and oracle $\mathsf{PCA}(w, \cdot)$ on queries (hp', v') from \mathcal{A} outputs 1 if $v' = \mathsf{PHash}(x, w, hp')$ and 0 otherwise.

SPHF for $\mathcal{L}^{\mathsf{Com}}$ using 'Double Pedersen' Commitment. Recall the trapdoor commitment scheme $(\mathsf{GPG},\mathsf{PG},\mathsf{Com},...)$ shown in Construction 4.1. Let $\mathsf{PG}_{\mathsf{sphf}}$ be the global parameter generation algorithm GPG in that construction, in particular $\mathsf{PG}_{\mathsf{sphf}}(1^{\kappa})$ outputs $\pi = (\mathbb{G}, q, g_1, g_2)$. Note that $\mathcal{L}^{\mathsf{Com}}[\pi]$ instantiated for this scheme contains tuples $(\overline{\pi} = (h_1, h_2), m, c = (c_1, c_2))$ s.t. there exists $d \in \mathbb{Z}_q$ s.t. $(c_1/h_1^m, c_2/h_2^m) = (g_1^d, g_2^d)$, i.e. if $dlog(g_1, c_1/h_1^m) = dlog(g_2, c_2/h_2^m)$. Define $\mathcal{L}_{pre}[\pi]$ as the set of pairs $(\overline{\pi}, m) \in \mathbb{G}^2 \times \mathbb{Z}_q$. We show that the SPHF for discrete-log equality [24] is an SPHF for $\mathcal{L}^{\mathsf{Com}}$ which is covert and OW-PCA if group G is uniformly encodable and GapDH assumption holds on G.

Construction 4.2 The SPHF for $\mathcal{L}^{\mathsf{Com}}$ proceeds as follows:

- $\mathsf{HKG}(\pi)$ picks $hk = (hk_1, hk_2) \leftarrow_{\scriptscriptstyle R} \mathbb{Z}_q^2$ and sets $hp = (g_1)^{hk_1} (g_2)^{hk_2}$ - $\mathsf{Hash}((\overline{\pi}, c, m), hk)$ parses $\overline{\pi} = (h_1, h_2)$ and $c = (c_1, c_2)$ and outputs

$$v \leftarrow (c_1/h_1^m)^{hk_1} (c_2/h_2^m)^{hk_2}$$

- PHash($(\overline{\pi}, c, m), d, hp$) outputs $v \leftarrow hp^d$

Theorem 4.2. SPHF in construction 4.2 is (1) covert if \mathbb{G} is uniformly encodabable and (2) OW-PCA under GapDH on \mathbb{G} , for $\mathsf{PG}_{\mathsf{sphf}}$ and $\mathcal{L}_{\mathsf{pre}}$ defined above.

Proof. Correctness is by inspection. SPHF is smooth on space $\mathcal{H}[\pi] = \mathbb{G}$ because this space is of the required super-polynomial size, and if $c_1 = g_1^d \cdot h_1^m$ and $c_2 = g_2^{d'} \cdot h_2^m$ for $d \neq d'$ then $v = (c_1/h_1^m)^{hk_1}(c_2/h_2^m)^{hk_2} = (g_1)^{d \cdot hk_1}(g_2)^{d' \cdot hk_2}$ is uniform in \mathbb{G} and independent of $hp = (g_1)^{hk_1}(g_2)^{hk_2}$. SPHF covertness follows because hp is a random element in $S[\pi] = \mathbb{G}$, and \mathbb{G} is uniformly encodable.

We argue that under GapDH assumption on G this SPHF is also OW-PCA secure. Note that for any $\pi, \overline{\pi}, m$, the OW-PCA challenge c, hp satisfies that (1) hp is random in \mathbb{G} and (2) $c = (c_1, c_2)$ s.t. $(c_1/h_1^m, c_2/h_2^m)$ is distributed as (g_1^d, g_2^d) for $d \leftarrow_{\mathbb{R}} \mathbb{Z}_q$. Note also that the PCA oracle can be implemented with a DDH oracle, which given query (hp', v') answers 1 iff $(g_1, c_1/h_1^m, hp', v')$ is a DDH tuple. Therefore, if an efficient adversary can solve for $v = hp^d$ given access to the PCA oracle, the reduction can break GapDH: Given a CDH challenge (g_1, hp, g_1^d) the reduction can set g_2, g_2^d by exponentiating g_1, g_1^d to a random value in \mathbb{Z}_q , and for arbitrary $\overline{\pi} = (h_1, h_2)$ and m, it can set (c_1, c_2) as $(g_1^d h_1^m, g_2^d h_2^m)$, and implement the PCA oracle using the DDH oracle as described above. \square

Covert Identity Escrow 4.3

We describe a Covert Identity Escrow (IE) scheme, an essential ingredient in our group cAKE construction of Section 6.

IE Syntax. An Identity Escrow (IE) scheme [42] is an entity authentication scheme with operational assumptions and privacy properties similar to a group signature scheme [22]. Namely, a designated party called a group manager (GM) uses a key generation algorithm KG to first generate a group public key qpk and a master secret key msk. Then, using the master secret key and a certificate generation algorithm CG, the group manager can issue each group member a membership certificate *cert* together with membership validity witness v. This pair allows a group member to authenticate herself as belonging to the group, but this authentication is anonymous in that multiple authentication instances conducted by the same party cannot be linked. In other words, the verifier is convinced that it interacts with some group member, in possession of some valid membership certificate, but it cannot tell which one. Following [13] we use the Verifier-Local Revocation (VLR) model for IE/group signature, where algorithm CG produces also a revocation token rt corresponding to certificate cert, and the authentication between a prover holding (gpk, cert, v) and the verifier holding gpk and a set of revocation tokens RTset is defined by a triple of algorithms CertBlind, Ver, Link, as follows:

- 1. The prover uses a certificate blinding algorithm CertBlind to create a blinded certificate bc from its certificate *cert*, and sends bc to the verifier.
- 2. The prover proves knowledge of witness v corresponding to the blinded certificate bc using a zeroknowledge proof of knowledge for relation

$$\mathcal{R}^{\mathsf{IE}} = \{ ((gpk, bc), v) \text{ s.t. } \mathsf{Ver}(gpk, bc, v) = 1 \}$$
(2)

3. The verifier accepts if and only if the above proof succeeds and the tracing algorithm Link does not link the blinded certificate to any revocation token in set RTset, i.e. if Link(gpk, bc, rt) = 0 for all $rt \in RTset$.

The IE syntax and correctness requirements are formally captured as follows:¹⁷

Definition 4.6. An identity escrow *(IE)* scheme is a tuple of efficient algorithms (KG, CG, CertBlind, Ver, Link) with the following syntax:

- Key Generation alg. KG picks a public key pair, $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa})$
- Certificate Generation alg. CG generates a certificate cert, its validity witness v, and revocation token $rt, (cert, v, rt) \leftarrow CG(msk)$
- Blinding alg. CertBlind outputs a blinded certificate, $bc \leftarrow CertBlind(cert)$
- Verification alg. Ver, s.t. if $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa}), (cert, v, rt) \leftarrow \mathsf{CG}(msk), and bc \leftarrow \mathsf{CertBlind}(cert), then \mathsf{Ver}(gpk, bc, v) = 1$
- Tracing alg. Link, s.t. if $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa}), (cert, v, rt) \leftarrow \mathsf{CG}(msk), and bc \leftarrow \mathsf{CertBlind}(cert), then \mathsf{Link}(gpk, bc, rt) = 1$

IE Security. Below we state the standard IE security properties [42], strengthened by covertness needed for our group cAKE construction.

The IE unforgeability property is that the adversary who receives some set of certificates, cannot create pair (bc, v) which satisfies the verification equation, i.e. Ver(gpk, bc, v) = 1, but which the tracing algorithm Link fails to link to the revocation tokens corresponding to the certificates received by the adversary. In the group cAKE application an adversary, in addition to holding some set of compromised certificates, can also observe revocation tokens and blinded certificates corresponding to non-compromised certificates. The definition below captures this by giving the adversary an arbitrary number of revocation tokens rt and certificates cert from which it can generate blinded certificates on its own:

Definition 4.7. We call an IE scheme unforgeable if for any efficient algorithm \mathcal{A} the probability that b = 1 in the following game is negligible in κ , for m, n polynomial in κ s.t. m < n:

- 1. set $b \leftarrow 0$ and $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa})$
- 2. for $i \in [1, n]$ set $(cert_i, v_i, rt_i) \leftarrow \mathsf{CG}(msk)$
- 3. $(bc^*, v^*) \leftarrow \mathcal{A}(gpk, \{cert_i, v_i, rt_i\}_{i \in [1,m]}, \{cert_i, rt_i\}_{i \in [m+1,n]})$
- 4. $b \leftarrow 1$ if $\operatorname{Ver}(gpk, bc^*, v^*) = 1$ and $\operatorname{Link}(gpk, bc^*, rt_i) = 0$ for all $i \in [1, m]$

(In the above game, tuples $(cert_i, v_i, rt_i)$ for $i \in [1, m]$ represent compromised certificates, set $\{rt_i\}_{i \in [m+1,n]}$ contains all additional revocation tokens the adversary learns, and set $\{cert_i\}_{i \in [m+1,n]}$ can be used to derive all blinded certificates the adversary receives from non-compromised parties.)

The IE covertness property strengthens the standard IE property of authentication anonymity [42]. Authentication anonymity asks that an adversary cannot link blinded certificate bc and decide e.g. whether they are generated from the same certificate or not. Covertness strengthens this by requiring that blinded certificates are indistinguishable from random elements in a uniformly encodable domain (hence they can be covertly encoded, see Section 2.4). Since each blinded certificate is indistinguishable from random domain element, it follows in particular that they are unlinkable. Similarly as in the unforgeability property, the adversary should be able to observe other certificates, hence in the definition below we hand the adversary the master secret key msk from which it can generate certificates, blinded certificates, and revocation tokens.

¹⁷ More generally, CertBlind should take witness v along with *cert* as input, and produce output v' along with *bc* as output, where v' is a validity witness for the *blinded certificate bc*. We use simpler syntax assuming that v' = v because it declutters notation, and it suffices for IE instantiation from Pointcheval-Sanders signatures [51].

Definition 4.8. We call an IE scheme covert if there is a uniformly encodable domain D s.t. for any efficient algorithm \mathcal{A} quantity $|p_0 - p_1|$ is negligible in κ for n, m polynomial in κ , where $p_b = \Pr[b' = 1]$ in the following game:

1. $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa})$

- 2. for $i \in [1, n]$ set $(cert_i, v_i, rt_i) \leftarrow \mathsf{CG}(msk)$
- 3. for all $(i, j) \in [1, n] \times [1, m]$:
- $if b = 1 \ then \ set \ bc_{ij} \leftarrow \mathsf{CertBlind}(cert_i) \ else \ pick \ bc_{ij} \leftarrow_{R} D$
- 4. $b' \leftarrow \mathcal{A}(msk, gpk, \{bc_{ij}\}_{i \in [1,n], j \in [1,m]})$

We require that the zero-knowledge proof for relation $\mathcal{R}^{\mathsf{IE}}$ in eq. (2) used is (based on) a Σ -protocol. We need this property to build a covert CKEM for the same relation using the Σ -to-CKEM compiler of Section 5.2.

Definition 4.9. We call an IE scheme Σ -protocol friendly if relation $\mathcal{R}^{\mathsf{IE}}$, Equation (2), admits a Σ -protocol with a uniformly encodable response space S_z .

Finally, we require IE to satisfy that the same blinded certificate cannot, except for negligible probability, correspond to two different honestly generated revocation tokens created on behalf of two different groups. This property allows the AKE scheme constructed in Section 6 to realize the group cAKE functionality \mathcal{F}_{g-cAKE} of Section 3, which assumes that if the real-world adversary attempts to authenticate using some group certificate then this implies a unique choice of a certificate, and hence also a group for which it was generated.

Definition 4.10. We call IE scheme unambiguous if:

(1) the probability that $\text{Link}(gpk_0, bc, rt_0) = \text{Link}(gpk_1, bc, rt_1) = 1$ is at most negligible for any efficient \mathcal{A} , where $(msk_b, gpk_b) \leftarrow \text{KG}(1^{\kappa})$, $(v_b, cert_b, rt_b) \leftarrow \text{CG}(msk_b)$ for $b \in \{0, 1\}$, and $bc \leftarrow \mathcal{A}(msk_0, v_0, cert_0, rt_0, msk_1, v_1, cert_1, rt_1)$;

(2) the same holds if the above experiment is adjusted by setting $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa})$ and $(v_b, cert_b, rt_b) \leftarrow \mathsf{CG}(msk)$ for $b \in \{0, 1\}$, and we measure the probability that $\mathsf{Link}(gpk, bc, rt_0) = \mathsf{Link}(gpk, bc, rt_1) = 1$.

Covert IE Instantiation using Pointcheval-Sanders Signature. We show how the Pointcheval-Sanders group signature [51] gives rise to a secure, covert, and Σ -protocol friendly IE scheme. In the following we will refer to this Covert IE implementation as *Pointcheval-Sanders IE* (PS IE).

Construction 4.3 Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear pairing of type 3 with $g(\hat{g})$ a generator of \mathbb{G}_1 (\mathbb{G}_2). The Pointcheval-Stern IE scheme works as follows:

- $\mathsf{KG}(1^{\kappa})$ picks $x, y \leftarrow_{\mathbb{R}} \mathbb{Z}_p$ and outputs msk = (x, y) and $gpk = (\hat{X}, \hat{Y}) = (\hat{g}^x, \hat{g}^y)$ (the public key includes also the description of the pairing group)
- $\mathsf{CG}(msk)$ picks $\tilde{\sigma} \leftarrow_{\scriptscriptstyle R} \mathbb{G}_1$, $v \leftarrow_{\scriptscriptstyle R} \mathbb{Z}_p$, sets $\tilde{\omega} \leftarrow \tilde{\sigma}^{x+y \cdot v}$, and outputs certificate cert = $(\tilde{\sigma}, \tilde{\omega})$, validity witness v, and revocation token $rt = \hat{Y}^v$
- CertBlind(cert = $(\tilde{\sigma}, \tilde{\omega})$) picks $t \leftarrow_{\mathbb{R}} \mathbb{Z}_p$ and outputs $bc = (\sigma, \omega) = (\tilde{\sigma}^t, \tilde{\omega}^t)$
- $-\operatorname{Ver}(gpk = (\hat{g}, \hat{X}, \hat{Y}), bc = (\sigma, \omega), v) = 1 \text{ iff } e(\sigma, \hat{X} \cdot \hat{Y}^v) = e(\omega, \hat{g})$
- $-\operatorname{Link}(gpk = (\hat{g}, \hat{X}, \hat{Y}), bc = (\sigma, \omega), rt) = 1 \text{ iff } e(\sigma, \hat{X} \cdot rt) = e(\omega, \hat{g})$

Theorem 4.3. Pointcheval-Sanders IE scheme, Construction 4.3, is unforgeable, covert, Σ -protocol friendly, and unambiguous, where unforgeability holds under Assumption 1, see Section 2.2, and covertness holds under the DDH assumption in \mathbb{G}_1 if group \mathbb{G}_1 is uniformly encodable.

Proof. We argue each of the above properties in turn:

Unforgeability. Unforgeability follows from the unforgeability of the Pointcheval-Sanders (PS) signature, proven under the same assumption [51]. In the PS signature scheme, tuples (v, cert) are formed as above, and they have the same security property. Note that there is no essential difference between certificate $cert = (\tilde{\sigma}, \tilde{\omega})$ and its blinding $bc = (\sigma, \omega)$: For a given validity witness v and group public key $gpk = (\hat{g}^x, \hat{g}^y)$, both cert and bc are distributed as pairs $(\sigma, \sigma^{x+y\cdot v})$ for random $\sigma \leftarrow \mathbb{G}_1$. The

security of PS signatures rests on the fact that under Assumption 1, Sec. 2.2, an adversary who gets tuples $(cert_i, v_i, rt_i)$ cannot create $(cert^*, v^*)$ s.t. $Ver(gpk, cert^*, v^*) = 1$ and $Link(gpk, cert^*, rt_i) = 0$ for all *i*. Moreover, this holds even if the adversary is also given any number of additional $(cert_j, rt_j)$ pairs. In more detail, notice that for each bc^* there is a unique v^* which satisfies the verification equation in Construction 4.3, implying that the adversary cannot produce a forgery for $i \in [1, m]$. Partition the event that the adversary wins into two cases. First, assume (bc^*, v^*) correspond to a $(cert_i, rt_i)$ for $i \in [m+1, n]$. Such adversary can be used to solve DLOG on \mathbb{G}_2 given $(h, h^{x+v^*y} \in \mathbb{G}_1^2$. The Link equation in Construction 4.3 implies that the only remaining case is that the forgery corresponds to a new v^* unseen by the adversary, but this happens with only negligible probability by Assumption 1.

Covertness. First, note that if \mathbb{G}_1 is uniformly encodable then so is domain $(\mathbb{G}_1)^2$ of blinded certificates. Secondly, each blinded certificate is distributed as (σ, σ^t) where $t = x + y \cdot v$. Since each v is random, so is each t, even to an adversary who holds msk = (x, y). Therefore, distinguishing between b = 0 and b = 1 in the covertness game for PS IE, is equivalent to distinguishing, for each $i \in [1, n]$, between (1) tuples of the form $\{(\sigma_{ij}, \sigma_{ij}^{t_i})\}_{j \in [1,m]}$ where $t_i \leftarrow_{\mathbb{R}} \mathbb{Z}_p$ and $\sigma_{i1}, ..., \sigma_{im} \leftarrow_{\mathbb{R}} \mathbb{G}_1$, and (2) m pairs of random \mathbb{G}_1 elements. The indistinguishability between the two follows from the DDH assumption on \mathbb{G}_1 .

 Σ -protocol. Consider relation $\mathcal{R}^{\mathsf{IE}}$, eq. (2), instantiated for the PS IE scheme:

$$\mathcal{R}^{\mathsf{PS-IE}} = \{ ((gpk = (\hat{g}, \hat{X}, \hat{Y}), bc = (\sigma, \omega)), v) \text{ s.t. } e(\sigma, \hat{X} \cdot \hat{Y}^v) = e(\omega, \hat{g}) \}$$
(3)

The Σ -protocol for $\mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$ is a tuple ($\mathsf{P}_1, \mathsf{P}_2, \mathsf{VRec}, S_{ch}, S_z$), see Section 2.3 for notation, where $S_{ch} = S_z = \mathbb{Z}_p$, and algorithms $\mathsf{P}_1, \mathsf{P}_2, \mathsf{VRec}$ are defined as follows, where gpk and bc are always parsed as $gpk = (\hat{g}, \hat{X}, \hat{Y})$ and $bc = (\sigma, \omega)$:

- $-\mathsf{P}_1((gpk, bc), v)$ picks $r \leftarrow \mathbb{Z}_p$ and sets $a \leftarrow e(\sigma, \hat{Y}^r)$
- $-\mathsf{P}_2((gpk, bc), v, r, ch)$ outputs $z = r + ch \cdot v \pmod{p}$
- Verifier accepts on x = (gpk, bc) and (a, ch, z) iff $a = e(\sigma, \hat{X}^{ch} \hat{Y}^z) \cdot e(\omega, \hat{g}^{-ch})$

The completeness and response uniqueness properties can be seen by inspection. For special HVZK, note that for any $ch \in \mathbb{Z}_p$, choosing $z \leftarrow \mathbb{Z}_p$ and setting $a \leftarrow e(\sigma, \hat{X}^{ch} \hat{Y}^z) \cdot e(\omega, \hat{g}^{-ch})$ generates identical transcript to that generated by an honest prover. For special strong soundness, note that the challenge space $S_{ch} = \mathbb{Z}_p$ is super-polynomial, furthermore for any (gpk, bc) and any two accepting transcripts (a, ch, z) and (a, ch', z') s.t. $ch' \neq ch$, the verification equations imply $e(\sigma, \hat{Y}^{\delta_z} \hat{X}^{\delta_{ch}}) = e(\omega, \hat{g}^{\delta_{ch}})$ for $\delta_{ch} = (ch - ch')$ and $\delta_z = (z - z')$, which implies that $e(\sigma, \hat{Y}^{\delta_z/\delta_{ch}} \hat{X}) = e(\omega, \hat{g})$, hence an efficient witness extractor can compute $v = \delta_z/\delta_{ch} \pmod{p}$ s.t. $((gpk, bc), v) \in \mathcal{R}^{\mathsf{PS-IE}}$. Finally, note that the prover's response space $S_z = \mathbb{Z}_p$ is uniformly encodable, see Sec. 2.4.

Unambiguity. For part (1), note that $\operatorname{Link}(gpk, bc, rt) = 1$ for $gpk = (\hat{g}, \hat{X}, \hat{Y})$ and $bc = (\sigma, \omega)$ iff $e(\sigma, \hat{X} \cdot rt) = e(\omega, \hat{g})$, i.e. iff $\operatorname{dlog}(\sigma, \omega) = \operatorname{dlog}(\hat{g}, \hat{X} \cdot rt)$. For a random group key gpk and random (cert, v, rt) generated for that group, we have $\hat{X} = \hat{g}^x$ and $rt = (\hat{Y})^v = \hat{g}^{y \cdot v}$ where x, y, v are all random in \mathbb{Z}_p . Therefore if \mathcal{A} on inputs random $msk_0 = (x_0, y_0)$, $msk_1 = (x_1, y_1)$, and random v_0, v_1 (from which \mathcal{A} can generate everything else it can see), generates bc s.t. $\operatorname{Link}(gpk_0, bc, rt_0) = \operatorname{Link}(gpk_1, bc, rt_1) = 1$, it must hold that $\operatorname{dlog}(\hat{g}, \hat{X}_0 \cdot rt_0) = \operatorname{dlog}(\hat{g}, \hat{X}_1 \cdot rt_1)$, i.e. that $x_0 + y_0 \cdot v_0 = x_1 + y_1 \cdot v_1$, which can happen with probability only 1/p. As for part (2), note that $\operatorname{Link}(gpk, bc, rt_0) = \operatorname{Link}(gpk, bc, rt_1) = 1$ iff $rt_0 = rt_1$, which holds with probability 1/p.

5 Covert Strong Simulation-Sound Conditional KEM

Conditional Key Encapsulation Mechanism (CKEM) [39] is a KEM counterpart of Witness Encryption (WE) [32] and Conditional Oblivious Transfer (COT) [25]. A CKEM for an efficiently verifiable relation \mathcal{R} (and a corresponding NP language $\mathcal{L}_{\mathcal{R}}$) is a protocol that allows sender S and receiver R, to establish, on input a statement x, a secure key K if R holds a witness w s.t. $(x, w) \in \mathcal{R}$. Since CKEM is an encryption counterpart to a zero-knowledge proof, we follow [39,9,40] and use ZKP terminology referring to CKEM properties, e.g. we call CKEM sound if S's output K_S is pseudorandom if $x \notin \mathcal{L}_{\mathcal{R}}$, and we call it strong sound [39] if w is extractable from any algorithm distinguishing K_S from random.

Benhamouda et al. [9] strengthened the notion of CKEM (called *Implicit Zero-Knowledge* therein) to include *simulatability*, i.e. that there exists an efficient simulator which for any $x \in \mathcal{L}_{\mathcal{R}}$ computes R's

output K_R without the knowledge of witness w for x, and simulation-soundness, i.e. that adversarial CKEM instances remain sound even in the presence of a simulator which simulates CKEM instances performed on behalf of honest players. Jarecki [40] extended simulation-sound CKEM of [9] to covertness, i.e. indistinguishability of a simulation (and hence also the real receiver) from a random beacon.

Here we adopt the covert zero-knowledge and simulation-sound CKEM notion which follows the above chain of works, but we modify it in several ways. First, we combine strong soundness of [39] and simulation-soundness of [9] to strong simulation-soundness, i.e. we require an efficient extractor that extracts a witness from an attacker who distinguishes S's output key from random on instance x in the presence of a simulator which plays the receiver's role on any instance $x' \neq x$. This is motivated by the group cAKE application where a reduction must extract a certificate forgery from an attacker who breaks sender's security of CKEM on a statement corresponding to a non-revoked certificate.

Our second change is introducing a *postponed-statement zero-knowledge* property to CKEM, which asks that there exists a postponed-statement simulator which simulates the CKEM on behalf of the receiver, i.e. recovers the same key K_R which an honest receiver would compute, not only without knowing the witness but also *without knowing the statement* used by the real-world receiver R, except after all CKEM messages are exchanged, i.e. in the final key-computation step of the receiver. This property is crucial in an application like group cAKE, because in the ideal-world group cAKE scheme, see the group cAKE functionality in Section 3, the simulator does not know the group to which a simulated party belongs. Indeed, the simulator does not even know if a party whose execution it simulates is a real party which executes the group cAKE for some group or it is a random beacon. Therefore, the simulator will not know the statement x on which the real-world party performs the CKEM, except in the final step in the case that (1) the adversary performs a CKEM for some group, and (2) the functionality confirms that the honest party involved in this execution is a real-world receiver R (and not a random beacon) and R runs on the same group the adversary does. At this point the simulator reconstructs the correct statement x the real-world R would have used in that case, and passes x to the *postponed-statement CKEM simulator* to compute R's output K_R .

The third change is that we cannot use proof *labels*, which were used to separate between honest and adversarial proof/CKEM instances in e.g. [40]. This change stems from the fact that whereas in many applications protocol instances can be tied to unique identifiers of participating parties, we cannot do so in the case of covert authentication. Indeed, an adversary \mathcal{A} interacting with a covert authentication system could forward statement x from receiver R to sender S, and forward S's CKEM for x from S to R. If in the simulation-soundness game \mathcal{A} learns R's output K_R then \mathcal{A} can trivially distinguish S's output K_S from random, as K_S and K_R are equal. Since this attack scenario corresponds to the case of AKE attacker who forwards protocol messages between R and S, we will handle that case separately as *eavesdropper security*, while in the simulation-soundness game we impose a restriction that the challenge \mathcal{A} -S interaction transcript differs from all \mathcal{A} -R transcripts. Note that both the relation $\mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$ for which we need this CKEM, and the SPHF tool we use to construct the CKEM scheme below, are malleable, e.g. if the adversary changes statement $x = (\sigma, \omega)$ to $x' = (\sigma^{\delta}, \omega^{\delta})$ then $x' \in \mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$ if $x \in \mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$. However, we obtain sufficient separation between CKEM instances by deriving the CKEM key via a random oracle (RO) hash on the SPHF-derived key and an interaction transcript.

In Section 5.1 below we define the covert zero-knowledge strong simulation-sound CKEM, and then in Section 5.2 we show a CKEM construction which achieves this covert CKEM notion in ROM for any relation \mathcal{R} with a Σ -protocol.

5.1 Definition of covert CKEM with strong simulation-soundness

Definition 5.1. A conditional key encapsulation mechanism (*CKEM*) for relation \mathcal{R} is an algorithms tuple (GPG, Snd, Rec) s.t. parameter generation $\mathsf{GPG}(1^{\kappa})$ generates *CRS* parameter π , and the sender Snd and receiver Rec are interactive algorithms which run on local respective inputs (π, x) and (π, x, w) , where each of them outputs a session key K as its local output. *CKEM* correctness requires that for all $(x, w) \in \mathcal{R}$ and $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$, if K_S, K_R are respective outputs of $\mathsf{Snd}(\pi, x)$ and $\mathsf{Rec}(\pi, x, w)$ interacting with each other, then $K_S = K_R$.

In the definition below we use the notation $\mathsf{P}_{\&Out}(x)$ for an interactive algorithm P that runs on input x and attaches its local output to its last message. (In our case this output will be a CKEM key K_S or K_R .) For notation $\mathsf{P}^{\$(\kappa)}$ refer to Section 2.4.

Definition 5.2. A CKEM for relation \mathcal{R} is covert zero-knowledge and strong simulation-sound if there exist efficient algorithms TGPG and psTGPG which on input 1^{κ} output parameters π together with trapdoor td, and interactive algorithms TRec and psTRec which runs on input (π , x, td), which satisfy the following properties:

- 1. Setup Indistinguishability: parameters π generated by $\mathsf{GPG}(1^{\kappa})$, $\mathsf{TGPG}(1^{\kappa})$, and $\mathsf{psTGPG}(1^{\kappa})$, are computationally indistinguishable.
- 2. Zero-Knowledge: For any efficient \mathcal{A} ,

$$\{\mathcal{A}^{\mathsf{RecO}(\pi,\cdot)}(\pi)\} \approx_c \{\mathcal{A}^{\mathsf{TRecO}(\pi,td,\cdot)}(\pi)\}$$

for $(\pi, td) \leftarrow \mathsf{TGPG}(1^{\kappa})$, where oracle $\mathsf{RecO}(\pi, \cdot)$ runs $\mathsf{Rec}_{\&\mathsf{Out}}(\pi, x, w)$ and $\mathsf{TRecO}(\pi, td, \cdot)$ runs $\mathsf{TRec}_{\&\mathsf{Out}}(\pi, x, td)$, on any query $(x, w) \in \mathcal{R}$ sent by \mathcal{A} .

- 3. Statement-Postponed Zero-Knowledge: The above property must hold for (psTGPG, psTRec) replacing (TGPG, TRec) where psTRec computes all its network messages given (π, td) and only uses x for its local output.
- 4. Receiver Covertness: For any efficient \mathcal{A} , $\{\mathcal{A}^{\mathsf{Rec}(\pi,x,w)}(st)\} \approx_c \{\mathcal{A}^{\mathsf{Rec}^{\$(\kappa)}}(st)\}$ for $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$ and $(x, w, st) \leftarrow \mathcal{A}(\pi)$ s.t. $(x, w) \in \mathcal{R}$.
- 5. Sender Covertness: For any efficient \mathcal{A} , $\{\mathcal{A}^{\mathsf{Snd}(\pi,x)}(st)\} \approx_c \{\mathcal{A}^{\mathsf{Snd}^{\$(\kappa)}}(st)\}$ for $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$ and $(st,x) \leftarrow \mathcal{A}(\pi)$.
- 6. Passive Security: For any efficient \mathcal{A} ,

$$\{\mathcal{A}(\pi, st, \mathtt{tr}, K_S)\} \approx_c \{\mathcal{A}(\pi, st, \mathtt{tr}, K')\}$$

for $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$, $(x, w, st) \leftarrow \mathcal{A}(\pi)$ s.t. $(x, w) \in \mathcal{R}$, $(\mathsf{tr}, K_S, K_R) \leftarrow [\mathsf{Snd}(\pi, x) \leftrightarrow \mathsf{Rec}(\pi, x, w)]$, $K' \leftarrow \{0, 1\}^{\kappa}$.

7. Strong Simulation-Soundness: There exists an efficient algorithm Ext s.t. for any deterministic efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, if $\epsilon = |p_0 - p_1|$ is non-negligible, then so is ϵ' , for p_b for b = 0, 1 and ϵ' defined as follows:

$$\begin{aligned} p_b &= \Pr\left[b' = 1: (\pi, td, x, st) \leftarrow \mathsf{Init}[\mathcal{A}_1](1^\kappa), b' \leftarrow \mathsf{Exp}_b[\mathcal{A}_2](\pi, td, x, st)\right] \\ \epsilon' &= \Pr\left[(x, w) \in \mathcal{R}: (\pi, td, x, st) \leftarrow \mathsf{Init}[\mathcal{A}_1](1^\kappa), w \leftarrow \mathsf{Ext}^{\mathcal{A}_2(st)}(\pi, td, x, st)\right] \end{aligned}$$

where

-
$$\operatorname{Init}[\mathcal{A}_1](1^{\kappa})$$
 sets $(\pi, td) \leftarrow \mathsf{TGPG}(1^{\kappa})$ and $(x, st) \leftarrow \mathcal{A}_1^{\mathsf{I}\operatorname{Rec}_{\&\operatorname{Out}}(\pi, \cdot, td)}(\pi);$

- $\operatorname{\mathsf{Exp}}_b[\mathcal{A}_2](\pi, td, x, st) \text{ outputs } b' = \mathcal{A}_2^{\operatorname{\mathsf{SndMod}}_{\operatorname{\&Out}}(b, \pi, x), \operatorname{\mathsf{TRec}}_{\operatorname{\&Out}}(\pi, \cdot, td)}(st) \text{ s.t.}$
 - SndMod_{&Out} $(1, \pi, x)$ runs Snd_{&Out} (π, x) ;
 - SndMod_{&Out} $(0, \pi, x)$ runs Snd (π, x) and then sends $K'_S \leftarrow \{0, 1\}^{\kappa}$;

Moreover, Exp_b rejects if \mathcal{A}_2 makes the transcript of an interaction with $\mathsf{SndMod}(b, \pi, x)$ the same as that of any interaction with $\mathsf{TRec}(\pi, x, td)$.

Discussion. The most direct comparison to the above notion of covert CKEM is a covert CKEM defined in [40]. Differences from [40] include (1) lack of labels, (2) strengthening of simulation-soundness to strong simulation-soundness, and (3) requirement that the CKEM facilitates statement-postponed simulation. Furthermore, (4) we allow the adversary in the strong simulation-soundness game to interact with the receiver even on the same statement x used in the challenge sender interaction, with the only constraint of excluding the trivial attack when the adversary passes all messages between S and R, i.e. when some \mathcal{A} -R transcript equals the \mathcal{A} -S transcript. We compensate for the latter constraint with (5) a *passive security* requirement, i.e. that if the adversary passes messages between S and R then the security holds even if the attacker knows the authentication tokens these parties use.

5.2 Compiler from Σ -protocol to covert CKEM in ROM

Our covert CKEM protocol, shown in Figure 3, is a compiler which creates a covert CKEM for relation \mathcal{R} from any Σ -protocol for \mathcal{R} . The two other tools this protocol requires are a *covert perfectly-binding trapdoor commitment* scheme, see Section 4.1, and a *covert and OW-PCA secure SPHF* for language $\mathcal{L}^{\mathsf{Com}}[\pi]$ associated with this commitment scheme, see Section 4.2 and equation (1). In addition, the compiler uses the ROM, and in particular it assumes that the commitment scheme has *RO-compatible instance parameters*, see Section 4.1, and it instantiates the instance parameter generation of the commitment with an RO hash $\mathsf{H}_{\mathsf{Com}}$. Usage of ROM is motivated by the goal of realizing all CKEM security properties at low cost in computation, communication, and round complexity. In particular, our CKEM has minimal round complexity: *one simultaneous flow*. It is natural to ask whether the same strong properties can be achieved, and at what costs, in the standard model, but we leave this as an open question.

Protocol Ingredients (see text):

- Σ -protocol ($\mathsf{P}_1, \mathsf{P}_2, \mathsf{VRec}, S_{ch}, S_z$) for relation \mathcal{R} ;
- covert perfectly-binding trapdoor commitment (GPG, PG, Com, Decom) on msg. space \mathcal{M} , with RO-compatible instance parameters with param. space \mathcal{C} ;
- covert SPHF (HKG, Hash, PHash) for $\mathcal{L}^{\mathsf{Com}}[\pi]$;
- CRH H and RO's H_1, H_2, H_{Com} with ranges resp. $\mathcal{M}, S_{ch}, \{0, 1\}^{\kappa}$, and \mathcal{C} ;

The GPG algorithm is the same as in the commitment scheme, i.e. $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$.

Rec.1: On inputs $(x, w) \in \mathcal{R}$, compute:

$$(a, r) \leftarrow \mathsf{P}_1(x, w)$$
$$\overline{\pi} \leftarrow \mathsf{H}_{\mathsf{Com}}(x)$$
$$(c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a))$$
$$ch \leftarrow \mathsf{H}_1(x, c)$$
$$z \leftarrow \mathsf{P}_2(x, w, r, ch)$$

and send (c, z) to S (using covert encoding).

Snd.1: S precomputes $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$ and sends hp to R (covertly encoded). Rec.2: Given hp, compute:

$$v_R \leftarrow \mathsf{PHash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), d, hp)$$

 $K_R \leftarrow \mathsf{H}_2(x, c, z, hp, v_R)$

Snd.2: On input x and given (c, z), compute:

 $ch \leftarrow \mathsf{H}_{1}(x, c)$ $\overline{\pi} \leftarrow \mathsf{H}_{\mathsf{Com}}(x)$ $a \leftarrow \mathsf{VRec}(x, ch, z)$ $v_{S} \leftarrow \mathsf{Hash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), hk)$ $K_{S} \leftarrow \mathsf{H}_{2}(x, c, z, hp, v_{S})$



Comparison with [39]. Our CKEM construction is a modification of the Σ -to-CKEM compiler of Jarecki [39], where (1) the commitment scheme Com which R uses to compute c in step R.1 must be a *trapdoor* commitment, where the commitment parameters are derived by an RO hash of the statement x, (2) the covert SPHF has an additional property of OW-PCA security, see Def. 4.5 in Sec. 4.2, and (3) the CKEM key output is not the SPHF hash value itself, but the RO hash of that value together with the language statement and the protocol transcript. Intuitively, the first change allows the CKEM to achieve statement-postponed zero-knowledge, since the trapdoor receiver can create a commitment without knowing the instance parameter $\overline{\pi}$. The second change assures security against a passive attacker. The last change allows for a stronger version of simulation-soundness, see Def. 5.2, which asks that the

Sender CKEM challenge is secure in the presence of Receiver CKEM oracle that can be executed *even on* the same statement, and the only restriction is that the CKEM transcripts of the adversary's interactions with the Sender and the Receiver cannot be the same. (The case of same transcripts is covered by the passive security property.) The proof of the following theorem is included in Appendix A:

Theorem 5.1. CKEM for \mathcal{R} shown in Fig. 3 is covert zero-knowledge and strong simulation-sound in ROM, if \mathcal{R} has a Σ -protocol with uniformly encodable response space S_z , trapdoor commitment Com is perfectly binding and covert, H is a CRH, and SPHF for \mathcal{L}^{Com} is covert, smooth, and OW-PCA secure.

Let $(p, g, \hat{g}_1, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be type-3 curve. Given $gpk = (\hat{X}, \hat{Y}) \in (\mathbb{G}_2)^2$ [assume gpk defines all curve parameters] and $bc = (\sigma, \omega) \in (\mathbb{G}_1)^2$, define:

$$\mathcal{R}^{\mathsf{PS}-\mathsf{IE}} = \{ ((gpk, bc), v) \,|\, e(\sigma, \hat{X} \cdot \hat{Y}^v) = e(\omega, \hat{g}) \}$$

Let (\mathbb{G}, q) be a DDH group, e.g. a standard curve. Let H be a CRH, and $\mathsf{H}_{\mathsf{Com}}, \mathsf{H}_1, \mathsf{H}_2$ be RO's, with ranges resp. $\mathbb{Z}_q, \mathbb{G}^2, \mathbb{Z}_q$, and $\{0, 1\}^{\kappa}$. Messages (c_1, c_2, z) and hp below are sent using covert encodings on \mathbb{G} and \mathbb{Z}_q . PG: On 1^{κ} , set $\pi = (g_1, g_2) \leftarrow \mathbb{G}^2$ (assume \mathbb{G} is chosen for sec. par. κ) Rec.1: R on x = (gpk, bc) and v picks $(r, d) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_q$, sets $a \leftarrow e(\sigma, \hat{Y}^r), \overline{\pi} = (h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}(x), (c_1, c_2) \leftarrow (g_1^d \cdot h_1^{\mathsf{H}(a)}, g_2^d \cdot h_2^{\mathsf{H}(a)}), z \leftarrow r + \mathsf{H}_1(x, c_1, c_2) \cdot v \mod p$, and sends (c_1, c_2, z) to S Snd.1: S precomputes $hk = (hk_1, hk_2) \leftarrow \mathbb{Z}_q^2$ and sends $hp = (g_1)^{hk_1}(g_2)^{hk_2}$ to R Rec.2: On message hp, R sets $v_R \leftarrow (hp)^d$ and $K_R \leftarrow \mathsf{H}_2((gpk, bc), c_1, c_2, z, hp, v_R)$ Snd.2: On statement x = (gpk, bc) and message (c_1, c_2, z) , S sets $\overline{\pi} = (h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}(x), a' \leftarrow e(\sigma, \hat{X}^{ch} \hat{Y}^z) \cdot e(\omega, \hat{g}^{-ch})$ for $ch = \mathsf{H}_1(x, c_1, c_2)$, sets $v_S \leftarrow (c_1 \cdot h_1^{-\mathsf{H}(a')})^{hk_1} \cdot (c_2 \cdot h_2^{-\mathsf{H}(a')})^{hk_2}$,

and $K_S \leftarrow \mathsf{H}_2((gpk, bc), c_1, c_2, z, hp, v_S)$



Efficient Instantiation. In Figure 4 we show an efficient instantiation of the generic CKEM construction from Fig. 3, for relation $\mathcal{R}^{\mathsf{PS-IE}}$ defined by Covert IE using Pointcheval-Sanders signatures, see Sec. 4.3, the "Double Pedersen" trapdoor commitment, see Sec. 4.1, and the associated SPHF, see Sec. 4.2. By the security of protocol building blocks (Theorems 4.1, 4.2, 4.3), we obtain a corollary of Theorem 5.1:

Corollary 1. *CKEM for* $\mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$ *in Fig. 4 is covert zero-knowledge and strong simulation-sound in ROM, if DDH and GapDH assumptions hold on group* \mathbb{G} , H *is a CRH, and group* \mathbb{G} *is uniformly encodable.*

6 Construction of Group Covert AKE Protocol

In Figures 5 and 6 we show algorithms (KG, CG, Auth) which implement a generic group cAKE construction from covert Identity Escrow (IE) and covert CKEM. In Figure 5 we show the group initialization algorithm KG and certificate generation algorithm CG, which implement respectively the Glnit and CertInit interfaces of UC group cAKE, as defined in Section 3. Figure 5 also shows the "input-retrieval" step in the implementation of the NewSession command, which triggers the online authentication algorithm Auth. The algorithm Auth itself, executing between two parties, is shown in Figure 6. Note that if a party is called with command (NewSession, ssid, \perp) then it executes as a random beacon, as noted in Figure 5, instead of following the Auth protocol of Figure 6.

The authentication protocol Auth in Figure 6 uses the same combination of IE and CKEM as in the covert AKE of [39], i.e. each party commits to its IE certificate, and then performs a CKEM to (implicitly and covertly) prove that it knows a valid secret key issued by the group manager, corresponding to this committed certificate. (Also, similarly as in [39], since the IE supports verifier-local revocation, each party uses algorithm Link to locally verify the committed certificate against each revocation token on its revocation list.) In spite of reusing the same construction paradigm, the novel aspects of this protocol are as follows: First, thanks to stronger CKEM properties we can show that this generic protocol realizes UC

Protocol Ingredients: – covert IE scheme IE = (KG, CG, CertBlind, Ver, Link) – covert CKEM scheme $\mathsf{CKEM} = (\mathsf{GPG}, \mathsf{Snd}, \mathsf{Rec})$ for relation $\mathcal{R}^{\mathsf{IE}}$ secure channel between GM and each party PCommon Reference String Generation: Sample CKEM global common reference string $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$ Initialization by GM on (Glnit, gid): Sample $(msk, gpk) \leftarrow_{\mathbb{R}} \mathsf{KG}(1^{\kappa})$, broadcast $(\mathsf{gid}, \mathsf{GM}, gpk, \pi)$ to all P's Certificate Generation by party P on (CertInit, gid, cid): P retrieves (gid, GM, gpk, π), sends cid to GM on a secure channel GM sets $(v, cert, rt) \leftarrow CG(msk)$, sends (v, cert, rt) to P on a secure channel P stores (gid, cid, v, cert, rt), GM stores $T_{rt}[gid, cid] \leftarrow rt$ Authenticated Key Exchange by party P on (NewSession, ssid, gid, cid, RTcids): 1. P retrieves $(\mathsf{gid}, \mathsf{GM}, gpk, \pi)$ and $(\mathsf{gid}, \mathsf{cid}, v, cert, \cdot)$ P sends (gid, RTcids) to GM on a secure channel GM sends back $\mathsf{RTset} = \{T_{rt}[\mathsf{gid}, \mathsf{cid}] \text{ s.t. } \mathsf{cid} \in \mathsf{RTcids}\}$ on a secure channel 2. P runs protocol Auth $((gpk, \pi), (v, cert), \mathsf{RTset})$ in Figure 6 P outputs (NewKey, ssid, K, cid_{CP}) for (K, cid_{CP}) output by protocol Auth Random Beacon implemented by party P on (NewSession, ssid, \perp): Party P runs Auth^{\$(κ)}, a random beacon of the same bandwidth as protocol Auth

Fig. 5. Generic group cAKE: Initialization and UC interface.

$\label{eq:point} \ensuremath{P} \ensuremath{\text{ on }} ((gpk,\pi),(v,\mathit{cert}),RTset)$	P' on $((gpk, \pi), (v', cert'), RTset')$
$bc \leftarrow CertBlind(cert) \qquad \underline{bc}$	$\bullet bc' \qquad bc' \leftarrow CertBlind(cert')$
$K_S \leftarrow Snd(\pi, (gpk, bc')) \qquad \longleftrightarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{array}{c} \hline \text{CKEM} \\ \textbf{P} \rightarrow \textbf{P}' \end{array} \longleftrightarrow \qquad Rec(\pi, (gpk, bc'), v') \rightarrow K'_R \\ \end{array}$
$K_R \leftarrow Rec(\pi, (gpk, bc), v) \qquad \longleftrightarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\overrightarrow{\text{CKEM}} \longleftrightarrow \qquad \text{Snd}(\pi, (gpk, bc)) \to K'_S$
If $\exists rt \in RTset s.t.$	If $\exists rt' \in RTset'$ s.t.
Link(gpk, bc', rt) = 1	Link(gpk, bc, rt') = 1
then $K \leftarrow \bot$, $cid_{CP} \leftarrow cid[rt]$	then $K' \leftarrow \bot$, $cid'_{CP} \leftarrow cid[rt']$
else $K \leftarrow H(\{K_S, K_R\}_{\mathrm{ord}}), cid_{CP} \leftarrow \bot$	else $K' \leftarrow H(\{K'_S, K'_R\}_{\mathrm{ord}}), cid_{CP} \leftarrow \bot$
H is RO with range $\{0,1\}^{\kappa}$, notation $\{a,b\}_{\text{ord}}$ stands for $(\min(a,b), \max(a,b))$	

Fig. 6. Generic group cAKE: protocol Auth, using covert encodings for bc/bc'.

group cAKE notion defined in Section 3. This implies that the protocol remains covert and secure under concurrent composition, e.g. that leakage of keys on any session does not endanger either covertness or security of any other session. Secondly, the strong notion of CKEM allows for minimal interaction, i.e. both receiver and sender can send only one message without waiting for their counterparty. Consequently, the generic Auth protocol in Figure 6 has a minimally-interactive instantiation shown in Figure 7.

The security of the above group cAKE construction is captured in the following theorem, with a proof included in Appendix B:

Theorem 6.1. Protocol $\Pi = (KG, CG, Auth)$ in Figs. 5,6 realizes UC Covert Authenticated Key Exchange if IE is secure, covert, and Σ -protocol friendly, and CKEM is covert zero-knowledge and strong simulation-sound.

Efficient Instantiation. Figure 7 shows a concrete instantiation of the generic group cAKE scheme shown in Figures 5,6. This instantiation uses the IE scheme based on Pointcheval-Sanders signatures, see Section 4.3, and the CKEM from Section 5 instantiated as shown in Figure 4. Note that the protocol has

We use the same parameters as CKEM for $\mathcal{R}^{\mathsf{PS-IE}}$ in Fig. 4, i.e. $(p, g, \hat{g}_1, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a type-3 curve, (\mathbb{G}, q) is a DDH group, and H is a CRH, and H_{Com} , H_1 , H_2 are RO's, with ranges resp. \mathbb{Z}_q , \mathbb{G}^2 , \mathbb{Z}_q , and $\{0,1\}^{\kappa}$. Values $(\sigma, \omega), (c_1, c_2, z), hp$ below are sent using covert encodings on \mathbb{G}_1, \mathbb{G} and \mathbb{Z}_q . Common Reference String Generation: Set $\pi = (g_1, g_2) \leftarrow_{\mathbf{R}} (\mathbb{G})^2$ Group Manager GM Initialization: Set $msk = (x, y) \leftarrow_{\mathbf{R}} (\mathbb{Z}_p)^2$ and $gpk = (\hat{X}, \hat{Y}) \leftarrow (\hat{g}^x, \hat{g}^y)$ Certificate Generation by GM: For P: Pick $(v, u) \leftarrow_{\mathbf{R}} (\mathbb{Z}_p)^2$, set $(\tilde{\sigma}, \tilde{\omega}) \leftarrow (g^u, g^{u(x+y \cdot v)})$, $rt \leftarrow \hat{Y}^v$ For P': Pick $(v', u') \leftarrow_{\mathbf{R}} (\mathbb{Z}_p)^2$, set $(\tilde{\sigma}', \tilde{\omega}') \leftarrow (g^{u'}, g^{u'(x+y \cdot v')})$, $rt' \leftarrow \hat{Y}^{v'}$ Authentication Protocol P on $((gpk, \pi), (v, (\tilde{\sigma}, \tilde{\omega})), \mathsf{RTset})$: P' on $((gpk, \pi), (v', (\tilde{\sigma}', \tilde{\omega}')), \mathsf{RTset}')$: $(\sigma', \omega') \leftarrow ((\tilde{\sigma}')^{t'}, (\tilde{\omega}')^{t'})$ for $t' \leftarrow_{\mathbf{R}} \mathbb{Z}_p$ $(\sigma, \omega) \leftarrow (\tilde{\sigma}^t, \tilde{\omega}^t)$ for $t \leftarrow_{\mathbf{R}} \mathbb{Z}_p$ $(c', \omega') \leftarrow ((c'), (\omega')') \text{ for } t' \leftarrow_{R} \mathbb{Z}_{2}$ $(r', d') \leftarrow_{R} \mathbb{Z}_{p} \times \mathbb{Z}_{q}, a' \leftarrow e(\sigma', \hat{Y}^{r'})$ $(h'_{1}, h'_{2}) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma', \omega')))$ $(c'_{1}, c'_{2}) \leftarrow (g_{1}^{d'} h_{1}^{\prime\mathsf{H}(a')}, g_{2}^{d'} h_{2}^{\prime\mathsf{H}(a')})$ $z' \leftarrow r' + ch' \cdot v' \mod p$ $(r, d) \leftarrow_{\mathrm{R}} \mathbb{Z}_p \times \mathbb{Z}_q, a \leftarrow e(\sigma, \hat{Y}^r)$ $(h_1,h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk,(\sigma,\omega)))$ $(c_1, c_2) \leftarrow (g_1^d h_1^{\mathsf{H}(a)}, g_2^d h_2^{\mathsf{H}(a)})$ $z \leftarrow r + ch \cdot v \bmod p$ for $ch = H_1(gpk, \sigma, \omega, c_1, c_2)$ for $ch' = \mathsf{H}_1(gpk, \sigma', \omega', c_1', c_2')$ $(hk_1, hk_2) \leftarrow_{\mathbf{R}} (\mathbb{Z}_q)^2$ $(hk'_1, hk'_2) \leftarrow_{\mathrm{R}} (\mathbb{Z}_q)^2$ $hp' \leftarrow g_1^{hk_1'} g_2^{hk_2'}$ $hp \leftarrow g_1^{hk_1}g_2^{hk_2}$ $bc' = (\sigma', \omega'), (c'_1, c'_2, z'), hp'$ $bc = (\sigma, \omega), (c_1, c_2, z), hp$ $K'_R \leftarrow \mathsf{H}_2(gpk, \sigma', \omega', c'_1, c'_2, z', (hp)^{d'})$ $K_R \leftarrow \mathsf{H}_2(gpk, \sigma, \omega, c_1, c_2, z, (hp')^d)$ $a' \leftarrow e(\sigma', \hat{X}^{ch'} \hat{Y}^{z'}) \cdot e(\omega', \hat{g}^{-ch'})$ $a \leftarrow e(\sigma, \hat{X}^{ch} \hat{Y}^z) \cdot e(\omega, \hat{g}^{-ch})$ for $ch' = \mathsf{H}_1(gpk, \sigma', \omega', c_1', c_2')$ for $ch = \mathsf{H}_1(gpk, \sigma, \omega, c_1, c_2)$ $(h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma, \omega)))$ $v'_S \leftarrow (c_1 h_1^{-\mathsf{H}(a)})^{hk'_1} \cdot (c_2 h_2^{-\mathsf{H}(a)})^{hk'_2}$ $\begin{array}{l} (h_1',h_2') \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk,(\sigma',\omega'))) \\ v_S \leftarrow (c_1'h_1^{-\mathsf{H}(a')})^{hk_1} \cdot (c_2'h_2^{-\mathsf{H}(a')})^{hk_2} \end{array}$ $K_S \leftarrow \mathsf{H}_2(gpk, \sigma', \omega', c_1', c_2', z', v_S)$ $K'_{S} \leftarrow \mathsf{H}_{2}(gpk, \sigma, \omega, c_{1}, c_{2}, z, v'_{S})$ If $\exists rt' \in \mathsf{RTset'}$ s.t. If $\exists rt \in \mathsf{RTset s.t.}$ $e(\sigma, \hat{X} \cdot rt) = e(\omega, \hat{g})$ $e(\sigma', \hat{X} \cdot rt) = e(\omega', \hat{g})$ then $K \leftarrow \bot$, $\mathsf{cid}_{\mathsf{CP}} \leftarrow \mathsf{cid}[rt]$ then $K' \leftarrow \bot$, $\operatorname{cid}_{\mathsf{CP}}' \leftarrow \operatorname{cid}[rt']$ else $K' \leftarrow \mathsf{H}(\{K'_S, K'_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \bot$ else $K \leftarrow \mathsf{H}(\{K_S, K_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \bot$ Output $(K, \operatorname{cid}_{\operatorname{CP}})$ Output $(K', \operatorname{cid}_{\operatorname{CP}}')$

Fig. 7. Instantiation of Covert AKE, with IE of Section 4.3 and CKEM of Figure 4

minimal interaction, as each party sends a single message without waiting for the counterparty, and it is quite practical: Its bandwidth is 6 group elements per party (2 in a base group of a type-3 elliptic curve and 4 in a standard group), and each party computes 10 fixed-base exp's, 4 variable-base (multi-)exp's, and 4 + n bilinear maps, where n is the size of the revocation list.

We include a step-by-step explanation why the code of player P in the Authentication Protocol in Fig. 7 is an instantiation of the generic group cAKE protocol of Figure 6 using the PS IE scheme of construction 4.3 and the CKEM instantiated as in Figure 4:

- The first line implements the certificate commitment procedure CertBlind;
- The next four lines implement step Rec.1 of $\mathsf{P} \leftarrow \mathsf{P}'$ CKEM, where P plays the receiver R on statement $x = (gpk, (\sigma, \omega))$ and witness v in $\mathcal{R}^{\mathsf{PS}-\mathsf{IE}}$;
- The next two lines lines implement step Snd.1 of the $P \rightarrow P'$ CKEM, where P plays the sender S for the counterparty's statement (not yet revealed);
- P's message contains the committed certificate $bc = (\sigma, \omega)$, R's P \leftarrow P' CKEM message $(c, z) = ((c_1, c_2), z)$, and S's P \rightarrow P' CKEM message hp;

- The first line of the code after receiving counterparty's message is step Rec.2, computing R's P \leftarrow P' CKEM key K_R ;
- The next four lines are step Snd.2, computing S's $P \to P'$ CKEM key K_S on statement x' = (gpk, bc') based on $bc' = (\sigma', \omega')$ received from P';
- The final four steps implements checking the counterparty's committed certificate against P's revocation list, and deciding on the output based on that, which is done as in the generic protocol in Figure 6.

References

- Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Topics in Cryptology – CT-RSA 2005, pages 191–208, 2005.
- Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy (S&P)*, pages 180–196, 2003.
- M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium* on Theory of Computing (STOC), page 419–428, 1998.
- Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Advances in Cryptology — ASIACRYPT 2001, pages 566–582. Springer, 2001.
- Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In CCS, pages 390–399. ACM, 2006.
- Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In CCS, pages 62–73. ACM, 1993.
- S.M. Bellovin and M. Merritt. Encrypted key-exchange: Password-based protocols secure against dictionary attacks. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84, 1992.
- Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Advances in Cryptology — EUROCRYPT, pages 274–285, 1993.
- Fabrice Benhamouda, Geoffroy Couteau, David Pointcheval, and Hoeteck Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In CRYPTO (2), volume 9216 of Lecture Notes in Computer Science, pages 107–129. Springer, 2015.
- Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.
- Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In CCS, pages 967–980. ACM, 2013.
- 12. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
- 13. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 168–177. ACM Press, October 2004.
- 14. Sean Bowe. Bls12-381: New zk-snark elliptic curve construction, Mar 2017.
- Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In CRYPTO, volume 6223 of Lecture Notes in Computer Science, pages 237–254. Springer, 2010.
- Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.
- 17. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Advances in Cryptology CRYPTO, pages 61–76, 2002.
- Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 1999.
- 19. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.
- Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In EUROCRYPT, volume 2332 of Lecture Notes in Computer Science, pages 337–351. Springer, 2002.
- Nishanth Chandran, Vipul Goyal, Rafail Ostrovsky, and Amit Sahai. Covert multi-party computation. In FOCS, pages 238–248. IEEE Computer Society, 2007.
- 22. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.

- Chongwon Cho, Dana Dachman-Soled, and Stanisław Jarecki. Efficient concurrent covert computation of string equality and set intersection. In RSA Conference on Topics in Cryptology - CT-RSA, page 164–179, 2016.
- 24. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.
- Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 1999.
- 26. Ivan Damgård. On ∑-protocols. https://cs.au.dk/~ivan/Sigma.pdf, 2010.
- Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.
- Karim Eldefrawy, Nicholas Genise, and Stanislaw Jarecki. Short concurrent covert authenticated key exchange (short cAKE). In Advances in Cryptology — ASIACRYPT 2023. Springer, 2023.
- Amos Fiat and Moni Naor. Broadcast encryption. In Advances in Cryptology CRYPTO' 93, pages 480–491. Springer, 1994.
- 30. Marc Fischlin. Trapdoor commitment schemes and their applications. PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany, 2001.
- Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. Discret. Appl. Math., 156(16):3113–3121, 2008.
- 32. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Symposium on Theory of Computing Conference, STOC'13, pages 467–476. ACM, 2013.
- Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In STOC, pages 197–206. ACM, 2008.
- Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In STOC, pages 365–377. ACM, 1982.
- Vipul Goyal and Abhishek Jain. On the round complexity of covert computation. In STOC, pages 191–200. ACM, 2010.
- Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. Khape: Asymmetric pake from key-hiding key exchange. In Advances in Cryptology – CRYPTO, pages 701–730, 2021.
- 37. Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. KHAPE: asymmetric PAKE from key-hiding key exchange. In CRYPTO (4), volume 12828 of Lecture Notes in Computer Science, pages 701–730. Springer, 2021.
- Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In CRYPTO, volume 2442 of Lecture Notes in Computer Science, pages 77–92. Springer, 2002.
- Stanislaw Jarecki. Practical covert authentication. In Public Key Cryptography, volume 8383 of Lecture Notes in Computer Science, pages 611–629. Springer, 2014.
- 40. Stanislaw Jarecki. Efficient covert two-party computation. In *Public Key Cryptography (1)*, volume 10769 of *Lecture Notes in Computer Science*, pages 644–674. Springer, 2018.
- Antoine Joux. A one round protocol for tripartite diffie-hellman. In ANTS, volume 1838 of Lecture Notes in Computer Science, pages 385–394. Springer, 2000.
- Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, Advances in Cryptology CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings, volume 1462 of Lecture Notes in Computer Science, pages 169–185. Springer, 1998.
- 43. Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for internet. In 1996 Internet Society Symposium on Network and Distributed System Security (NDSS), pages 114–127, 1996.
- 44. Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Advances in Cryptology CRYPTO 2003, pages 400–425. Springer, 2003.
- Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Advances in Cryptology CRYPTO 2005, pages 546–566, 2005.
- Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In CRYPTO, volume 3621 of Lecture Notes in Computer Science, pages 546–566. Springer, 2005.
- 47. Rajendra Kumar and Khoa Nguyen. Covert authentication from lattices. In ACNS, volume 13269 of Lecture Notes in Computer Science, pages 480–500. Springer, 2022.
- Mark Manulis, Benny Pinkas, and Bertram Poettering. Privacy-preserving group discovery with linear complexity. In International Conference on Applied Cryptography and Network Security - ACNS, pages 420–437, 2010.
- Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol, https://signal.org/docs/ specifications/x3dh/, 2016.
- Lan Nguyen. Accumulators from bilinear pairings and applications. In Topics in Cryptology CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings, pages 275–292, 2005.

- 51. David Pointcheval and Olivier Sanders. Short randomizable signatures. In CT-RSA, volume 9610 of Lecture Notes in Computer Science, pages 111–126. Springer, 2016.
- 52. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. J. Cryptol., 13(3):361–396, 2000.
- Phillip Rogaway. Nonce-based symmetric encryption. In FSE, volume 3017 of Lecture Notes in Computer Science, pages 348–359. Springer, 2004.
- 54. Jacob Appelbaum Roger Dingledine. How governments have tried to block tor. https://oldsite. andreafortuna.org/security/files/TOR/slides-28c3.pdf.
- hack-proof making 55. Anmol Sachdeva. Darpa ananonymous and mobile commu-Online FOSSBYTES nication system. Article. 2019.https://fossbytes.com/ darpa-anonymous-hack-proof-mobile-communication-system/.
- 56. Andrew Shallue and Christiaan E. van de Woestijne. Construction of rational points on elliptic curves over finite fields. In ANTS, volume 4076 of Lecture Notes in Computer Science, pages 510–524. Springer, 2006.
- Wazen M. Shbair, Thibault Cholez, Antoine Goichot, and Isabelle Chrisment. Efficiently bypassing sni-based https filtering. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 990–995, 2015.
- 58. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
- Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In *Financial Cryptography*, volume 8437 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2014.
- N S Vipin and M Abdul Nizar. Efficient on-line spam filtering for encrypted messages. In 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), pages 1–5, 2015.
- Luis von Ahn, Nicholas J. Hopper, and John Langford. Covert two-party computation. In STOC, pages 513–522. ACM, 2005.
- Riad S. Wahby and Dan Boneh. Fast and simple constant-time hashing to the BLS12-381 elliptic curve. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(4):154–179, 2019.

A Proof of security for the Covert CKEM construction

We include the proof of Theorem 5.1 deferred from Section 5.2, i.e. the proof of security of the Covert CKEM construction of Figure 3.

Proof. Trapdoor receiver algorithm TRec and psTRec are shown resp. in Figure 8 and 9. The trapdoor parameters generation TGPG for TRec is identical to the standard parameter generation GPG, and the only trapdoor assumed by the trapdoor receiver algorithm TRec, denoted td, is a control over the random oracle H₁. Likewise psTGPG for psTRec is identical to GPG, except its trapdoor td is a control over the random oracle H_{com}.

TRec(π , x, td), where td is the control of RO H₁: TRec.1 forms message (c, z) to S as follows: $(ch, z) \leftarrow_{\mathbb{R}} S_{ch} \times S_{z}$ $a \leftarrow \mathsf{VRec}(x, ch, z)$ $\overline{\pi} \leftarrow \mathsf{H}_{\mathsf{Com}}(x)$ $(c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a))$ $ch \to \mathsf{H}_{1}(x, c)$ (and abort if H₁ has been queried on (x, c) previously)

TRec.2, given S's message hp, computes K_R as Rec.2 does:

 $v_R \leftarrow \mathsf{PHash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), d, hp)$ $K_R \leftarrow \mathsf{H}_2(x, c, z, hp, v_R)$

Fig. 8. Trapdoor receiver TRec for CKEM of Fig. 3.

 $spTRec(\pi, td, x)$, where x is withheld till spTRec.2 and td is the control of RO H_{Com}: On adversary's new query x to H_{Com}:

$$(\overline{\pi}, tk) \leftarrow \mathsf{TPG}(\pi)$$

 $\overline{\pi} \to \mathsf{H}_{\mathsf{Com}}(x) \text{ (and associate } tk \text{ with } x)$

spTRec.1(π , td) forms message (c, z) to S as follows:

$$(c, td_{\mathsf{Com}}) \leftarrow \mathsf{TCom}(\pi)$$
$$z \leftarrow_{\mathsf{R}} S_z$$

spTRec.2, given x and S's message hp, retrieves tk associated with x, and sets:

 $ch \leftarrow \mathsf{H}_1(x, c)$ $a \leftarrow \mathsf{VRec}(x, ch, z)$ $d \leftarrow \mathsf{TDecom}(\pi, \overline{\pi}, c, tk, td_{\mathsf{Com}}, \mathsf{H}(a))$ $v_R \leftarrow \mathsf{PHash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), d, hp)$ $K_R \leftarrow \mathsf{H}_2(x, c, z, hp, v_R)$

Fig. 9. Statement-postponed trapdoor receiver spTRec for CKEM of Fig. 3.

Lemma A.1. CKEM for \mathcal{R} in Fig. 3 is Setup Indistinguishable in the ROM.

Proof. Setup indistinguishability is immediate in ROM, because TGPG and psTGPG are identical to GPG, and trapdoor td each of them generates denotes only a control over random oracle, resp. H₁ and H_{Com}.

Lemma A.2. CKEM for \mathcal{R} in Fig. 3 is Zero-Knowledge in the ROM assuming trapdoor commitment Com is covert, Com has RO-compatible commitment instance parameters, and that \mathcal{R} has a Σ -protocol.

Proof. Figure 8 depicts the trapdoor simulator TRec. The Zero-Knowledge property follows from the Σ -protocol's Special Honest Verifier Zero-Knowledge property (Section 2.3): that for all fixed $ch \in S_{ch}$, sampling $z \leftarrow S_z$ then $a \leftarrow \mathsf{VRec}(x, ch, z)$ is identically distributed to (a, z) computed in the honest manner: $(a, r) \leftarrow \mathsf{P}_1(x, w)$ and $z \leftarrow \mathsf{P}_2(x, w, r, ch)$. Randomizing over ch matches the distribution of ch in the ROM. Therefore, Rec and TRec have the same distribution conditioned on the event that TRec does not abort. Conversely, TRec only aborts if H_1 is queried twice on (x, c) and this happens with negligible probability because c is computationally indistinguishable from a uniformly encodable set of superpolynomial size, Definition 4.2.

Lemma A.3. CKEM for \mathcal{R} in Fig. 3 is Statement-Postponed Zero-Knowledge in the ROM assuming trapdoor commitment Com is covert, Com has RO-compatible commitment instance parameters, and that \mathcal{R} has a Σ -protocol.

Proof. Figure 9 depicts the trapdoor simulator spTRec. First note that the view of S is (c, z) and the output of the random oracles. The former, (c, z), in spTRec (Figure 9) is computationally indistinguishable from the real protocol because of Com's trapdoor covertness and the Σ -protocol's Special Honest Verifier Zero Knowledge property (Section 2.3): that for all fixed $ch \in S_{ch}$, sampling $z \leftarrow S_z$ then $a \leftarrow \mathsf{VRec}(x, ch, z)$ is identically distributed to (a, z) computed in the honest manner: $(a, r) \leftarrow \mathsf{P}_1(x, w)$ and $z \leftarrow \mathsf{P}_2(x, w, r, ch)$. Randomizing over ch matches the distribution of ch in the ROM. The other random oracle programming is done for $\overline{\pi}$ which is computationally indistinguishable from $\mathcal{U}(\mathcal{C})$ by Definition 4.2. Statement-Postponed Zero-Knowledge is clear from Figure 9.

Lemma A.4. *CKEM for* \mathcal{R} *in Fig. 3 is* receiver and sender covert *assuming the trapdoor commitment* scheme Com is covert, the SPHF for \mathcal{L}^{Com} is statistically covert, and \mathcal{R} 's Σ -protocol has a uniformly encodable response space S_z .

Proof. Receiver covertness follows from the covertness of the commitment scheme, c from $(c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a))$ is covert, and the covertness from Σ -protocol has a uniformly encodable response space S_z (covert encodings here are implicit). See Definition 4.2. Sender covertness follows immediately from Definition 4.4, that hp is covert (e.g., in a uniformly encodable space).

Lemma A.5. CKEM for \mathcal{R} in Fig. 3 is passively secure in the ROM if the SPHF for $\mathcal{L}^{\mathsf{Com}}$ is OW-PCA secure.

Proof. Note that for any $(x, w) \in \mathcal{R}$, the CKEM transcript $\mathsf{Snd}(\pi, x) \leftrightarrow \mathsf{Rec}(\pi, x, w)$ and key are generated as $tr \leftarrow (c, z, hp)$ and $K_S \leftarrow \mathsf{H}_2(x, c, z, hp, v)$ where (a, z) is generated by Σ -protocol prover on inputs (x, w) (and access to oracle H₁ which generates challenge ch), $(c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a))$, $(hk, hp) \leftarrow \mathsf{HKG}(\pi), \text{ and } v \leftarrow \mathsf{Hash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), hk).$

The reduction to OW-PCA security of SPHF for $\mathcal{L}^{\mathsf{Com}}$ recreates the above view by generating (a, z)using $(\mathsf{P}_1,\mathsf{P}_2,\mathsf{H}_1)^{(x,w)}$, and then using SPHF OW-PCA challenge $(\pi,\overline{\pi},c,hp)$ generated for message m = H(a). Since the only way an adversary can distinguish the real K_S key from random is by querying H_2 on (x, c, z, hp, v') for $v' = \mathsf{PHash}((\pi, \overline{\pi}, c, a), d, hp)$, the reduction monitors these queries, and identifies v' = v using the PCA oracle on (v', hp) to test v' in each H₂ query.

Lemma A.6. CKEM for \mathcal{R} in Fig. 3 is strong simulation-sound in the ROM if H is a CRH, the SPHF for $\mathcal{L}^{\mathsf{Com}}$ is statistically smooth and is OW-PCA secure, and that \mathcal{R} has a Σ -protocol.

Proof. Assume $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ s.t. $\mathsf{Adv}_{\mathcal{A}} = |p_1 - p_0|$ is non-negligible, where p_b for b = 1, 0 is defined as follows:

- $-p_1$ is the probability that $\mathcal{A}_2(st)$ outputs 1 on access to oracles $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x)$ and $\mathsf{TRec}_{\&\mathsf{Out}}(\pi, \cdot, td)$ for $(\pi, td) \leftarrow \mathsf{TPG}(1^{\kappa})$ and $(x, st) \leftarrow \mathcal{A}_1(\pi)$ on access to oracle $\mathsf{TRec}_{\&\mathsf{Out}}(\pi, \cdot, td)$, constrained so that oracle $Snd_{\&Out}$ is queried once and the A-S transcript cannot equal to any A-R transcript.
- $-p_0$ is the probability that $\mathcal{A}_2(st)$ outputs 1 in the same game as above except that oracle $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x)$ sets its output K as a random κ -bit string.

Recall that $\mathsf{TPG}(1^{\kappa})$ generates random π , and td denotes simulator TRec 's control over RO hash functions H_1, H_2 . For challenge bit b, an interaction of \mathcal{A} with sender S modeled by oracle Snd_{&Out}(π, \cdot) goes as follows:

- $-\mathcal{A}$ receives hp for $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$ from S
- \mathcal{A} sends (x, c, z) to S \mathcal{A} receives K_S^b from S, where $K_S^0 \leftarrow \{0,1\}^{\kappa}$ and $K_S^1 = \mathsf{H}_2(x, c, z, hp, v_S)$ for $\overline{\pi} \leftarrow \mathsf{H}_{\mathsf{Com}}(x), v_S = \mathsf{Hash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), hk)$ and $a = \mathsf{VRec}(x, \mathsf{H}_1(x, c), z)$.

Consider also the *i*-th interaction of \mathcal{A} with oracle $\mathsf{TRec}_{\&\mathsf{Out}}(\pi, \cdot, td)$:

- $-\mathcal{A}$ sends statement x_i to R
- $-\mathcal{A}$ receives (c_i, z_i) from R created as $(ch_i, z_i) \leftarrow S_{ch} \times S_z, a_i \leftarrow \mathsf{VRec}(x_i, ch_i, z_i), \overline{\pi} \leftarrow \mathsf{H}_{\mathsf{Com}}(x_i),$ $(c_i, d_i) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a_i)), \text{ and } \mathsf{H}_1(x_i, c_i) \leftarrow ch_i$
- (game aborts if H_1 was queried on this input before)
- \mathcal{A} sends to R projection key hp_i s.t. $(x_i, c_i, z_i, hp_i) \neq (x, c, z, hp)$
- \mathcal{A} receives $K_i = \mathsf{H}_2(x_i, c_i, z_i, hp_i, v_R)$ for $v_R = \mathsf{PHash}((\pi, \overline{\pi}, c_i, \mathsf{H}(a_i)), d_i, hp_i)$.

Denote the event that $(x_i, c_i) = (x, c)$ as EQ_i , let EQ be the union of events EQ_i over all sessions *i*, and let NEQ be the complement of EQ . Let $\mathsf{Adv}^E_{\mathcal{A}}$ stand for the adversarial advantage constrained by event *E*, i.e. where the above security game is modified so \mathcal{A} 's true output is ignored and artificially set to 0 if event *E* does not occur. Note that if $\mathsf{Adv}_{\mathcal{A}} \ge \epsilon$ then either $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EQ}} \ge \epsilon/2$ or $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NEQ}} \ge \epsilon/2$ because $\Pr[\mathsf{EQ} \cup \mathsf{NEQ}] = 1.$

Case 1: Consider first the case when $\operatorname{Adv}_{\mathcal{A}}^{\mathsf{NEQ}} \geq \epsilon/2$, i.e. when the attacker succeeds with non-negligible advantage without making any \mathcal{A} -R values (x_i, c_i) equal to (x, c) in the \mathcal{A} -S interaction. In this case the proof, based on the Bellare-Pointcheval forking lemma [52,5], is the same as in [39], and we sketch it here. Let $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ},j}$ be \mathcal{A} 's advantage restricted further by the constraint that (x, c) in \mathcal{A} -S interaction forms \mathcal{A} 's *j*-th query to oracle H₁. (If \mathcal{A} makes no such query we can add a wrapper that does when \mathcal{A} sends x, c, zto S.) If \mathcal{A} makes at most $q_{\rm H}$ queries to H₁ then there exists j s.t. $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ},j} \geq \epsilon/(2q_{\rm H})$. Let CS be the event that (x, c, z) implies a correct statement, i.e. that $(\pi, \overline{\pi}, c, {\rm H}(a)) \in \mathcal{L}^{\operatorname{Com}}$ for $a = \operatorname{VRec}(x, {\rm H}_1(x, c), z)$. If $\neg \operatorname{CS}$ then by SPHF smoothness value K_S^1 is distributed identically as K_S^0 , hence $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ},j,\operatorname{CS}} \geq \epsilon/(2q_{\rm H})$. (If SPHF smoothness is statistical then $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ},j,\operatorname{CS}}$ and $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ},j}$ can be apart by at most negligible difference.) The extractor Ext runs the beginning of the security game generating (π, td) as $\mathsf{TPG}(1^{\kappa})$

and providing access to TRec to \mathcal{A} until \mathcal{A} makes the *j*-th query (x, c) to H₁. If this query does not correspond to the challenge statement *x* and pair (c, z) which \mathcal{A} sends to S in the security game, Ext aborts. Otherwise Ext saves tuple (ch, z) for $ch = H_1(x, c)$. With probability at least $\epsilon' = \operatorname{Adv}_{\mathcal{A}}^{\operatorname{NEQ}, j, \operatorname{CS}}$ this tuple satisfies that H(*a*) for $a = \operatorname{VRec}(x, \operatorname{H}_1(x, c), z)$ is committed in *c*. Re-running the same experiment with (deterministic) \mathcal{A} but "forking" on the *j*-th query to H₁ by setting H₁(*x*, *c*) to a fresh random value ch', can lead to a second tuple (ch', z') s.t. H(*a'*) for $a' = \operatorname{VRec}(x, \operatorname{H}_1(x, c'), z')$ is committed in *c*, and by the standard forking lemma argument, this rewinding strategy succeeds with probability at least $\epsilon'' \geq (\epsilon')^2/2$. The probability that ch' = ch is only $1/|S_{ch}|$, which must be negligible, and if $a' \neq a$ we have a break in the CRH property of H, so with probability at least $\epsilon'' - (\delta_{\mathcal{A},\mathsf{H}} + 1/|S_{ch}|)$, where $\delta_{\mathcal{A},\mathsf{H}}$ is the negligible advantage of the implied reduction to CRH of H, we have a' = a and $ch' \neq ch$, and by the special strong soundness of Σ , there is an efficient extraction procedure which extracts w s.t. $(x, w) \in \mathcal{R}$ from two such transcripts.

Case 2: We will argue that the second case, i.e. $\operatorname{Adv}_{\mathcal{A}}^{\mathsf{EQ}} \geq \epsilon/2$, is not possible for non-negligible ϵ . This will imply that for non-negligible ϵ the first case is the only one possible, and thus complete the proof. Assume for contradiction that $\operatorname{Adv}_{\mathcal{A}}^{\mathsf{EQ}}$ is non-negligible. Since \mathcal{A} can start only polynomially-many sessions TRec sessions, there must be an i s.t. $\operatorname{Adv}_{\mathcal{A}}^{\mathsf{EQ}_i}$ is non-negligible, i.e. that \mathcal{A} has non-negligible advantage in a security game when $(x, c) = (x_i, c_i)$.

Case 2(a): Consider first the case when $z \neq z_i$. By the Σ -protocol response uniqueness property this implies that $a = \mathsf{VRec}(x, \mathsf{H}_1(x, c), z)$ and $a_i = \mathsf{VRec}(x, \mathsf{H}_1(x, c), z_i)$ satisfy $a \neq a_i$. By CRH property of H, there is at most negligible chance that $\mathsf{H}(a) = \mathsf{H}(a_i)$, hence we can assume that $\mathsf{H}(a) \neq \mathsf{H}(a_i)$. Since c_i is created by TRec as a commitment to $\mathsf{H}(a_i)$, by perfect binding property of Com the same value $c = c_i$ is not a commitment to $\mathsf{H}(a)$, which means that the challenge statement $(\pi, \overline{\pi}, c, \mathsf{H}(a))$ is not in $\mathcal{L}^{\mathsf{Com}}$. Therefore by SPHF smoothness the real and random CKEM key values K_S^1 and K_S^0 are distributed identically, hence in this case advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EQ}_i \cap [z \neq z_i]}$ has to be negligible.

Case 2(b): Consider the second case when $z = z_i$, which implies that $a = a_i$, and therefore statement $x^{\mathsf{Com}} = (\pi, \overline{\pi}, c, \mathsf{H}(a))$ is in $\mathcal{L}^{\mathsf{Com}}$. Let adversary's advantage in this case, $\epsilon = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{EQ}_i \cap [z=z_i]}$, be nonnegligible. Denote *i*-th session of TRec as $\mathsf{R}^{(i)}$. Note that \mathcal{A} gets hp from S for $(hp, hk) \leftarrow \mathsf{HKG}(1^{\kappa})$, and that since $(x, c, z) = (x_i, c_i, z_i)$ the adversary can pass to $\mathsf{R}^{(i)}$ only some value $hp_i \neq hp$. Observe that \mathcal{A} receives from S the CKEM challenge key K_S set as $K_S^1 = \mathsf{H}_2(x, c, z, hp, v_S)$ (or as $K_S^0 \leftarrow \{0, 1\}^{\kappa}$) for $v_S = \mathsf{Hash}(x^{\mathsf{Com}}, hk)$, and that \mathcal{A} receives from $\mathsf{R}^{(i)}$ the receiver's CKEM key $K_i = \mathsf{H}_2(x, c, z, hp_i, v_R)$ for $v_R = \mathsf{PHash}(x^{\mathsf{Com}}, d, hp_i)$, where (c, d) are honestly generated by $\mathsf{R}^{(i)}$ as $(c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, \mathsf{H}(a))$ for $a = \mathsf{VRec}(x, ch, z)$ and random ch, z (and setting $\mathsf{H}_1(x, c)$ to ch). (Note that we discounted the case that H_1 -setting can fail in the first security game.) In the Random Oracle model, if A has advantage ϵ in distinguishing K_S^1 from K_S^0 then \mathcal{A} must query H_2 on point (x, c, z, hp, v_S) with probability ϵ . Note also that \mathcal{A} does not learn anything from $\mathsf{R}^{(i)}$ output K_i unless it also queries H_2 on (x, c, z, hp_i, v_R) . Whether it does not perform that latter query, we can construct a reduction \mathcal{A}' to OW-PCA security of SPHF. Reduction \mathcal{A}' runs the security game above on OW-PCA challenge parameters π (note that TPG generates π in the same way as PG, and trapdoor td is solely a backdoor to RO's H_1, H_2 , setting hp output by S to the hp value in the OW-PCA challenge, and running $\mathsf{TRec}(\pi, \cdot, td)$ to service each \mathcal{A} -R interaction except for the *i*-th instance $\mathsf{R}^{(i)}$. For the latter \mathcal{A}' picks ch, z and sets $a = \mathsf{VRec}(x, ch, z)$ as TRec, sets the OW-PCA challenge message $m \leftarrow \mathsf{H}(a)$, receives c generated by the OW-PCA security game as $(c, d) \leftarrow \mathsf{Com}(\pi, \mathsf{H}(a))$, sets $\mathsf{H}_1(x, c) \leftarrow ch$ as TRec, and outputs (c, z) to \mathcal{A} . (Since we assume that \mathcal{A} passes this tuple to S we can denote it c, z instead of c_i, z_i .) Note that S's challenge SPHF value is $v_S = \mathsf{Hash}(x^{\mathsf{Com}}, hk)$ for $x^{\mathsf{Com}} = (\pi, \overline{\pi}, c, m)$, thus if \mathcal{A}' captures this v_S value from H₂ query (x, c, z, hp, v_S) of \mathcal{A} then \mathcal{A}' wins the OW-PCA game. However, \mathcal{A}' has no access to d so it cannot compute key $K_i = H_2(x, c, z, hp_i, v_R)$ for $v_R = \mathsf{PHash}(x^{\mathsf{Com}}, d, hp_i)$. This is why we need SPHF one-wayness to hold in the presence of the plaintext-checking oracle: Using oracle $\mathsf{PCA}(d, \cdot)$ reduction \mathcal{A}' can test, on every query (x, c, z, hp_i, v_R) of \mathcal{A} to H_2 , if $v_R = \mathsf{PHash}(x^{\mathsf{Com}}, d, hp_i)$, and if so then \mathcal{A}' sets $K_i \leftarrow \mathsf{H}_2(x, c, z, hp_i, v_R)$. This shows that \mathcal{A}' succeeds in the OW-PCA game with the same advantage which A has in CKEM simulation soundness if $(x, c, z) = (x_i, c_i, z_i)$. Since under OW-PCA assumption this advantage must be negligible, this completes the proof.

By combining Lemmas A.1, A.2, A.3, A.4, A.5, and A.6, the proof is complete.

33

B Proof of security for the group cAKE scheme

We include the proof of Theorem 6.1 deferred from Section 6, i.e. the proof of security of the group cAKE construction shown in Figures 5 and 6 in Section 6.

Note on the secure channels. The protocol in Figure 5 assumes a secure channel $P \leftrightarrow GM$ between the special party GM and each party P. This channel is secure in the sense of "lead pipe", i.e. the adversary does not even know if anyone transmits anything on it. For example, in the certificate generation P requests and then receives a group certificate (v, cert, rt) on this channel, but the adversary \mathcal{A} does not even learn that this communication takes place, hence in the ideal group cAKE model, see Figure 2. If the real-world adversary \mathcal{A} learns that certification takes place, this would be simulatable as well, if the idealworld adversary \mathcal{A}^* , a.k.a. the simulator, was told that P is getting some certificate from GM in response to the environment's certification generation query (CertInit, gid, cid) to P. However, when P runs the online authentication protocol, in response to the environment's query (NewSession, ssid, gid, cid, RTcids) to P, our implementation assumes that P uses the $P \leftrightarrow GM$ to exchange the set RTcids of the environment-level identifiers of revoked certificate to the set RTset of implementation-level revocation tokens rt generated for the corresponding certificates by GM. Even if this communication were secure, a standard channel would leak the number of the revoked certificates P uses to \mathcal{A} , which would be not simulatable unless functionality \mathcal{F}_{g-cAKE} leaks the same to the ideal-world adversary \mathcal{A}^* . Our notion of the $\mathsf{P} \leftrightarrow \mathsf{GM}$ in essence assumes that the issuance of certificates to honest parties and dissemination of the revocation tokens happen "out of bounds" for the real-world adversary \mathcal{A} . (Consequently, the ideal-world \mathcal{A}^* does not have to receive information related to these actions.) This appears to be a clean interface between the real-world covert authentication and our UC model \mathcal{F}_{g-cAKE} . In particular, if the environment wants to pass any information to the real-world adversary about either honest players' certificates or the size or content of their revocation lists, the environment can do this, and our results imply security in that case as well.

Simulator construction. Recall that, by definition 3.1, we need to show a construction of the idealworld adversary algorithm \mathcal{A}^* , i.e. the *simulator*, which on interaction with functionality \mathcal{F}_{g-cAKE} and the real-world adversary \mathcal{A} , emulates a view which no efficient environment \mathcal{Z} can distinguish from a real-world interaction between the same adversary \mathcal{A} and honest parties running protocol Π . Below we show the construction of \mathcal{A}^* for the proof that under the assumptions stated in Theorem 6.1 on IE and CKEM, no efficient \mathcal{Z} can distinguish between the real-real world interaction defined by Π and the ideal-world interaction defined by \mathcal{F}_{g-cAKE} and simulator \mathcal{A}^* shown below.

The ideal-world algorithm \mathcal{A}^* acts as follows:

- 1. At initialization \mathcal{A}^* sets $(\pi, td) \leftarrow \mathsf{TGPG}(1^{\kappa})$.
- 2. On any query (Glnit, GM, gid) from \mathcal{F}_{g-cAKE} , \mathcal{A}^* samples $(msk, gpk) \leftarrow \mathsf{KG}(1^{\kappa})$, saves (gid, msk, gpk) , and adds gpk to set Gset.
- 3. Note that \mathcal{A}^* does not learn from \mathcal{F}_{g-cAKE} when some certificate is created. However, if \mathcal{Z} sends a permission to \mathcal{A}^* to issue either (CompCert, P, gid, cid) or (RevealRT, P, gid, cid) queries to \mathcal{F}_{g-cAKE} for some (gid, cid), then whichever comes first, \mathcal{A}^* generates $(v, cert, rt) \leftarrow CG(msk)$, associates this certificate with (gid, cid), and reveals resp. (v, cert) (on CompCert) or rt (on RevealRT), and adds cid to lists CompCert^{gid} and RevealRT^{gid} (on CompCert) or only to list RevealRT^{gid} (on RevealRT), just like \mathcal{F}_{g-cAKE} does.
- When A* gets (NewSession, P, ssid, gid, cid) from F_{g-cAKE}, i.e. if P runs NewSession on (gid, cid) for cid ∈ RevealRT^{gid}, A* retrieves (v, cert) corresponding to (gid, cid) (note that A* must have created these if cid ∈ RevealRT^{gid}) and:
 - (a) \mathcal{A}^* sends $bc \leftarrow \mathsf{CertBlind}(cert)$ to \mathcal{A} and receives bc' from \mathcal{A} ;
 - (b) \mathcal{A}^* runs $K_S \leftarrow \mathsf{Snd}(\pi, (gpk, bc'))$, interacting with \mathcal{A} ;
 - (c) \mathcal{A}^* runs $K_R \leftarrow \mathsf{Rec}(\pi, (gpk, bc), v)$, interacting with \mathcal{A} ;
 - (d) Set tr[P, ssid] to the interaction transcript and $K^* \leftarrow H(\{K_S, K_R\}_{ord});$
 - (e) If tr[P, ssid] = tr[P', ssid'] for any P', ssid': then set $msg_{\mathcal{F}} \leftarrow (\text{Connect}, P, ssid, P', ssid')$ and reset $K^* \leftarrow \bot$;
 - (f) else if $\exists \operatorname{cid}^* \in \operatorname{Reveal}\mathsf{RT}^{\operatorname{gid}}$ associated with rt^* s.t. $\operatorname{Link}(gpk, bc', rt^*)=1$: then set $msg_{\mathcal{F}} \leftarrow (\operatorname{Impersonate}, \mathsf{P}, \operatorname{ssid}, \operatorname{gid}, \operatorname{cid}^*)$;
 - (g) else set $msg_{\mathcal{F}} \leftarrow (\text{Interfere}, \mathsf{P}, \mathsf{ssid})$ and reset $K^* \leftarrow \bot$;

(h) Send messages $msg_{\mathcal{F}}$ and (NewKey, P, ssid, K^*) to \mathcal{F}_{g-cAKE} .

- 5. When \mathcal{A}^* gets (NewSession, P, ssid, \perp) from \mathcal{F}_{g-cAKE} , i.e. if P is either a random beacon or it runs NewSession on (gid, cid) for cid \notin RevealRT^{gid}, then:
 - (a) \mathcal{A}^* samples random bc from CertBlind output domain and sends it to \mathcal{A} ;
 - (b) \mathcal{A}^* runs Snd.1(π) interacting with \mathcal{A} , saving its private state st_s ;
 - (c) \mathcal{A}^* runs psTRec.1(π , td) interacting with \mathcal{A} , saving its private state st_{R} ;
 - (d) When \mathcal{A} sends bc' and interactions end, set $tr[\mathsf{P}, ssid]$ to their transcript;
 - (e) If tr[P, ssid] = tr[P', ssid'] for any P', ssid': then set $msg_{\mathcal{F}} \leftarrow (Connect, P, ssid, P', ssid')$ and $K^* \leftarrow \bot$;
 - (f) else if $\exists (gid^*, cid^*)$ s.t. $cid^* \in \mathsf{RevealRT}^{gid^*}$ and $\mathsf{Link}(gpk^*, bc', rt^*) = 1$ for gpk^*, rt^* associated with $\mathsf{gid}^*, \mathsf{cid}^*: \mathsf{then set} \ x \leftarrow (gpk^*, bc') \text{ and do:}$
 - set $K_S \leftarrow \text{Snd.2}(\pi, x, st_S)$ and $K_R \leftarrow \text{psTRec.2}(\pi, td, x, st_R)$; reset $K^* \leftarrow H(\{K_S, K_R\}_{\text{ord}})$;

 - $\text{ set } msg_{\mathcal{F}} \leftarrow (\mathsf{Impersonate}, \mathsf{P}, \mathsf{ssid}, \mathsf{gid}^*, \mathsf{cid}^*);$
 - (g) else set $msg_{\mathcal{F}} \leftarrow (\mathsf{Interfere}, \mathsf{P}, \mathsf{ssid})$ and $K^* \leftarrow \bot$;
 - (h) Send messages $msg_{\mathcal{F}}$ and (NewKey, P, ssid, K^*) to \mathcal{F}_{g-cAKE} .

Notational conventions. We use P_{ssid} for an instance of party P running subsession ssid. Note that the environment \mathcal{Z} identifies group instances with identifiers gid and certificate and revocation token instances within a given group with identifiers cid, while in the implementation these correspond to respectively group public keys gpk, certificates (v, cert), and revocation tokens rt. To avoid the need to constantly translate between these two levels, we say that \mathcal{Z} "tells $\mathsf{P}_{\mathsf{ssid}}$ to run Auth on (gpk, cert, RTset)" if \mathcal{Z} sends (NewSession, ssid, gid, cid, RTcids) to P for gid corresponding to gpk, cid corresponding to cert, and revoked certificate identifier set RTcids corresponding to a set of revocation tokens RTset. We likewise say that \mathcal{Z} "tells $\mathsf{P}_{\mathsf{ssid}}$ to run $\mathsf{Auth}^{\$(\kappa)}$ " when \mathcal{Z} sends (NewSession, ssid, \bot) to P , and we say that " $\mathsf{P}_{\mathsf{ssid}}$ " outputs (K, rt), where K is either \perp or a key and rt is either \perp or a revocation token, if P outputs (NewKey, ssid, K, cid_{CP}) for cid_{CP} corresponding to rt. However, since a real session P_{ssid} always outputs (K, rt) s.t. $K = \perp$ if and only if $rt \neq \perp$, we simplify the syntax and assume that real $\mathsf{P}_{\mathsf{ssid}}$ outputs either K or rt (but not both), and it never outputs \perp , while a random beacon $\mathsf{P}_{\mathsf{ssid}}$ always outputs \perp . In the ideal-world execution, we say that "instance $\mathsf{P}_{\mathsf{ssid}}$ runs" some algorithm when the simulator \mathcal{A}^* runs this algorithm on behalf of $\mathsf{P}_{\mathsf{ssid}}$. We use RTset^{gpk} and CCset^{gpk} to denote the set of resp. revocation tokens and compromised certificates, corresponding to sets $\mathsf{RevealRT}^{\mathsf{gid}}$ and $\mathsf{CompCert}^{\mathsf{gid}}$ of resp. revoked and compromised cid's within group gid corresponding to qpk. Finally, in either the real-world or the ideal-world executions, we treat the real-world adversary \mathcal{A} as an interface of \mathcal{Z} .

Note on static compromise model. Recall that functionality $\mathcal{F}_{g\text{-cAKE}}$ imposes a static compromise model, see Section 3, therefore we can assume that \mathcal{Z} issues all certificate commands CompCert and all revocation token reveal commands RevealRT at the outset. This partitions the set of certificates into two sets, the "revealed", whose corresponding revocation token are in RTset^{gpk} and the "hidden", which are not. The set of revealed certificates is further divided between the "compromised", whose certificates were subject to the CompCert query, and "revealed-only", whose revocation tokens were subject to only the RevealRT query. All P_{ssid} sessions running on revealed certificates are simulated by \mathcal{A}^* in step 4 while sessions running on hidden certificates are simulated by \mathcal{A}^* in setp 5. The divergencies between the real world and the ideal world view which these two simulation methods create are summarized below, for the revealed certificates in step 3 and for the hidden certificates in step 4.

Summary of real-world vs. ideal-world differences. Recall that we need to argue that for any \mathcal{Z} (we adopt the standard convention that \mathcal{A} is only an interface of \mathcal{Z}),

$$\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{g-cAKE}},\mathcal{A}^*,\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}\approx_c \{\mathbf{Real}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}$$

We will show the above via a sequence of games below, but first we list the summary code of the ideal-world execution, in as much as it differs from the real-world execution. This walk-through of the differences between ideal-world and real-world is made with references to the group cAKE protocol shown in Fig. 6 in Section 6, the simulator \mathcal{A}^* shown above, and the functionality \mathcal{F}_{g-cAKE} in Fig. 2 in Section 3.

^{1.} π is generated by GPG in the real-world and by TGPG in the ideal-world.

- 2. If \mathcal{Z} tells $\mathsf{P}_{\mathsf{ssid}}$ to run $\mathsf{Auth}^{\$(\kappa)}$ then this is what \mathcal{Z} sees in the real world, while in the ideal world $\mathsf{P}_{\mathsf{ssid}}$ runs as \mathcal{A}^* in step 5, i.e. (s1) it sends random bc, (s2) on the outgoing CKEM it runs $\mathsf{Snd.1}(\pi)$, (s3) on the incoming CKEM it runs $\mathsf{psTRec}(\pi, td)$, (s4) when these complete and \mathcal{A} sends bc', \mathcal{A}^* performs some computation which results in sending $msg_{\mathcal{F}} \in \{\mathsf{Connect}, \mathsf{Impersonate}, \mathsf{Interfere}\}$ and NewKey to $\mathcal{F}_{\mathsf{g-CAKE}}$. However, if $\mathsf{P}_{\mathsf{ssid}}$ runs $\mathsf{Auth}^{\$(\kappa)}$ then $\mathcal{F}_{\mathsf{g-CAKE}}$ marks it random, hence (regardless of $msg_{\mathcal{F}}$) session $\mathsf{P}_{\mathsf{ssid}}$ outputs \bot in both the real and the ideal worlds. Thus the only difference between the two is in the network messages, i.e. steps (s1)-(s3) above.
- 3. If \mathcal{Z} tells $\mathsf{P}_{\mathsf{ssid}}$ to run Auth on $(gpk, cert, \mathsf{RTset})$ s.t. $rt \in \mathsf{RTset}^{gpk}$ for rt corresponding to cert, then this is what \mathcal{Z} sees in the real world, while in the ideal world $\mathsf{P}_{\mathsf{ssid}}$ also runs Auth on the above inputs when sending outgoing messages, but the local output of $\mathsf{P}_{\mathsf{ssid}}$ is determined as follows, depending on transcript $\mathsf{tr}[\mathsf{P},\mathsf{ssid}]$ and bc' received from \mathcal{A} :
 - (a) If tr[P, ssid] = tr[P', ssid'] (let gpk', cert', RTset' be the inputs of $P'_{ssid'}$ if it is a real session, and let rt' correspond to cert') then (c1) P_{ssid} outputs rt' if $P'_{ssid'}$ is real, gpk = gpk', and $rt' \in RTset$; (c2) P_{ssid} outputs the same key as $P'_{ssid'}$ if $P'_{ssid'}$ is real, it has output a key, gpk = gpk', and $rt' \notin RTset$; (c3) P_{ssid} outputs a random key in any other case.
 - (b) If tr[P, ssid] is adversarial and Link(gpk, bc', rt) = 0 for all $rt \in RTset^{gpk}$, then P_{ssid} outputs random key K;
 - (c) If tr[P, ssid] is adversarial and $\exists rt \in \mathsf{RTset}^{gpk}$ s.t. $\mathsf{Link}(gpk, bc', rt) = 1$: i. if $rt \in \mathsf{RTset}$ then P_{ssid} outputs rt;
 - ii. if $rt \in \mathsf{CCset}^{gpk} \setminus \mathsf{RTset}$ then $\mathsf{P}_{\mathsf{ssid}}$ outputs K^* set by \mathcal{A}^* in step 4d; iii. else P , outputs random key K:
 - iii. else $\mathsf{P}_{\mathsf{ssid}}$ outputs random key K;
- 4. If \mathcal{Z} tells $\mathsf{P}_{\mathsf{ssid}}$ to run Auth on $(gpk, cert, \mathsf{RTset})$ s.t. $rt \notin \mathsf{RTset}^{gpk}$ for rt corresponding to cert, then this is what \mathcal{Z} sees in the real execution, while in the ideal execution $\mathsf{P}_{\mathsf{ssid}}$ runs as in steps (s1)-(s3) in item 2 above, but when these complete and \mathcal{A} sends bc' then $\mathsf{P}_{\mathsf{ssid}}$ local output is as follows, depending on $tr[\mathsf{P}, \mathsf{ssid}]$ and bc' received from \mathcal{A} :
 - (a) If tr[P, ssid] = tr[P', ssid'] then P_{ssid} output is as in step 3a above;
 - (b) If tr[P, ssid] is adversarial and $Link(gpk^*, bc', rt) = 0$ for all gpk^* and $rt \in RTset^{gpk^*}$, then P_{ssid} outputs random key K;
 - (c) If tr[P, ssid] is adversarial and $Link(gpk^*, bc', rt) = 1$ for some gpk^* and $rt \in RTset^{gpk^*}$ then: i. If $gpk^* = gpk$ and $rt \in RTset$ then P_{ssid} outputs rt;
 - ii. If $gpk^* = gpk$ and $rt \in \mathsf{CCset}^{gpk} \setminus \mathsf{RTset}$ then $\mathsf{P}_{\mathsf{ssid}}$ outputs K^* set by \mathcal{A}^* in step 5f;
 - iii. else $\mathsf{P}_{\mathsf{ssid}}$ outputs random key K;

Game sequence. We fix an efficient environment algorithm \mathcal{Z} , which is a presumed distinguisher between the real-world and ideal-world interaction, and we show a sequence of game changes which bridges between \mathcal{Z} 's view of the real-world interaction defined by group cAKE scheme Π , which we denote Game 0, and the ideal-world interaction implied by functionality \mathcal{F}_{g-cAKE} and the simulator \mathcal{A}^* shown above, which we denote Game 10. For any *i*, we use Gi to denote the event that \mathcal{Z} outputs 1 in interaction with Game i, and our goal is to prove that $|\Pr[G0] - \Pr[G10]| \leq \mathsf{negl}(\kappa)$ under the assumptions stated in the theorem.

GAME 0 (real world): This is the real-world game as defined by protocol Π .

GAME 1 (key/rt transfer on passive session): We change the game so that it acts like in the ideal world in step 3a, i.e. if a real session $\mathsf{P}_{\mathsf{ssid}}$ is connected to some other session $\mathsf{P}'_{\mathsf{ssid}'}$ which is real (i.e. it is not a random beacon) then we don't compute the output of $\mathsf{P}_{\mathsf{ssid}}$ as in the real world, but we shortcut it in two cases, where gpk, cert, RTset and gpk', cert', RTset' are inputs of resp. $\mathsf{P}_{\mathsf{ssid}}$ and $\mathsf{P}'_{\mathsf{ssid}'}$ and rt' corresponds to cert': (1) if gpk = gpk' and $rt' \in \mathsf{RTset}$ then $\mathsf{P}_{\mathsf{ssid}}$ outputs rt', and (2) if gpk = gpk', $rt' \notin \mathsf{RTset}$, and $\mathsf{P}'_{\mathsf{ssid}'}$ has terminated with a key then $\mathsf{P}_{\mathsf{ssid}}$ outputs the same key which $\mathsf{P}'_{\mathsf{ssid}'}$. Note that after this change the game looks like the ideal-world game in step 3a in case of either clauses (c1) or (c2). Change (1) creates no difference in the game view by *IE correctness*, specifically that $\mathsf{Link}(gpk, bc', rt') = 1$ if $bc' \leftarrow \mathsf{CertBlind}(cert')$ for (cert', rt') generated by $\mathsf{CG}(msk)$. Change (2) is indistinguishable by *IE unambiguity*, specifically that the probability that $\mathsf{Link}(gpk, bc', rt) = 1$ for $bc' \leftarrow \mathsf{CertBlind}(cert')$ for (cert', rt') generated by $\mathsf{CG}(msk)$. Change (2) is indistinguishable by IE unambiguity, specifically that the probability that $\mathsf{Link}(gpk, bc', rt) = 1$ for $bc' \leftarrow \mathsf{CertBlind}(cert')$ for (cert', rt') generated by $\mathsf{CG}(msk)$ and any $rt \neq rt'$, and CKEM correctness, i.e. that if $\mathsf{P}_{\mathsf{ssid}}$ and $\mathsf{P}'_{\mathsf{ssid}}$ run CKEM on correct statement, witness pair then they get the same K, and hence the same key. Consequently, $|\Pr[\mathsf{G0}] - \Pr[\mathsf{G1}]| \leq \mathsf{negl}(\kappa)$.

GAME 2 (random key on passive session): We modify the game so it acts like the ideal world in step 3a, i.e. if a real session $\mathsf{P}_{\mathsf{ssid}}$ is connected to some other session $\mathsf{P}'_{\mathsf{ssid}'}$ and clauses (c1) and (c2) are not triggered, i.e. we are in the "otherwise" clause (c3), then instead of computing the output of $\mathsf{P}_{\mathsf{ssid}}$ as in the real world, we shortcut it by making $\mathsf{P}_{\mathsf{ssid}}$ output a random key $K \leftarrow_{\mathsf{R}} \{0,1\}^{\kappa}$. Note that after this change the game looks like the ideal-world game in step 3a.

First, note that if clauses (c1) and (c2) are not triggered then the probability that $\mathsf{P}_{\mathsf{ssid}}$ outputs some revocation token $rt \neq \bot$ (and $K = \bot$), is negligible: In the case that $\mathsf{P}'_{\mathsf{ssid}'}$ is random, then there is only negligible probability that $\mathsf{Link}(gpk, bc', rt) = 1$ for any $rt \in \mathsf{RTset}^{gpk}$ and bc' random in CertBlind's domain, because by *IE covertness* such bc' is indistinguishable from a bc' generated for an independently created *cert'*, and by *IE unambiguity* a revocation token rt corresponding to a random certificate *cert* will not work in Link on a blinded version of another random certificate *cert'*. In the second case, that $\mathsf{P}'_{\mathsf{ssid}'}$ is real but $gpk' \neq gpk$, then there is only negligible probability that $\mathsf{Link}(gpk, bc', rt) = 1$ for any rtcorresponding to *cert* generated for gpk and bc' output by CertBlind(*cert'*) for *cert'* generated for gpk': This follows from *IE unambiguity* and *IE correctness* because IE correctness implies that bc' has to be correct for gpk', hence IE unambiguity implies that it cannot be at the same time correct for gpk.

Second, if $\mathsf{P}_{\mathsf{ssid}}$ does not output $K \neq \bot$ then recall that it outputs its key as $K = \mathsf{H}(\{K_S, K_R\}_{\mathrm{ord}})$ for $K_S \leftarrow \mathsf{Snd}(\pi, (gpk, bc'))$ and $K_R \leftarrow \mathsf{Rec}(\pi, (gpk, bc), v)$. Since $tr[\mathsf{P}, ssid] = tr[\mathsf{P}', ssid']$, in the latter interaction Rec interacts with either honest party running (case 1) $Snd(\pi, (gpk, bc))$, or (case 2) $\mathsf{Snd}(\pi, (gpk', bc)),$ or (case 3) $\mathsf{Snd}^{\$(\kappa)}$. (The first two cases occur for real $\mathsf{P}'_{\mathsf{ssid}'}$ running on resp. gpk' = gpkand $gpk' \neq gpk$, while the third case occurs if $P'_{ssid'}$ is a random beacon.) We claim that in all of these cases the probability that the adversary queries H on K_R is negligible. Assume otherwise and let $\mathsf{P}_{\mathsf{ssid}}$ and $P'_{sid'}$ be a pair of instances for which this event occurs with non-negligible probability. Consider a modified interaction in which either case 2 or case 3 of $P'_{ssid'}$ are modified to case 1. By *CKEM sender* covertness, the change from case 2 to case 3 creates at most a negligible difference, and likewise the change from case 3 to case 1 creates at most a negligible difference in the probability of this event. Finally, assume that this event happens in case 1. That case leads to to contradiction by CKEM passive security, because the reduction would create a valid blinded certificate bc and witness v as inputs for $\mathsf{P}_{\mathsf{ssid}}$, and it would emulate the whole rest of the game except for distinguishing between a real $K_R = K_S$ output by $\mathsf{Snd}(\pi, (qpk, bc))$ and $\mathsf{Rec}(\pi, (qpk, bc), v)$ and a random key, because only the first one could be subject of \mathcal{A} 's query to H (except for the negligible probability). This concludes the argument that $|\Pr[\mathsf{G1}] - \Pr[\mathsf{G2}]| \le \mathsf{negl}(\kappa).$

Game reassessment: By the changes made above, Game 2 acts as the ideal-world in clause 3a. Note that it also follows the ideal-world game in clause 3(c)i, because in the real world P_{ssid} outputs rt because Link(gpk, bc', rt) = 1 for $rt \in RTset$, exactly like the ideal world in this clause. Furthermore, Game 2 also follows the ideal-world game in clause 3(c)ii, because if $rt \notin RTset$ then the ideal-world game outputs $K = H(\{K_S, K_R\}_{ord})$ for K_S, K_R computed as in the real-world game. We would like to argue that the game remains indistinguishable also if it starts acting like the ideal-world game in clauses 3b and 3(c)iii. Note that these cases correspond to an adversarial transcript, i.e. an active attack, in which bc' is either (clause 3b) not linked to any $rt \in RTset^{gpk}$ or (clause 3(c)ii) linked to some $rt \in RTset^{gpk}$ which is not in $RTset \cup CCset^{gpk}$. In other words, it is not on the revocation list which P_{ssid} uses and it is also not a compromised certificate (otherwise this would correspond to a successful compromise, as in clause 3(c)ii). We can unite these two clauses is a single clause, which outputs a random key if Link does not link bc' to any rt' in $RTset \cup CCset^{gpk}$. In other words, we want to argue that if the adversary uses a non-compromised certificate (randomized certificates are part of the protocol transcript so they can be replayed, and further randomized, by the adversary) which is not on P_{ssid} 's revocation list, then P_{ssid} will output a key which the adversary cannot tell from random.

Note that if adversary sends bc' and then computes K_R (by sending it as query to H) then by CKEM strong soundness there is an extractor which can extract witness v' s.t. $((gpk, bc'), v') \in \mathcal{R}^{\mathsf{IE}}$, i.e. $\mathsf{Ver}(gpk, bc', v') = 1$. If at the same time bc' avoids tracing with any $rt \in \mathsf{CCset}^{gpk}$, we would like to use tuple (bc', v') as an attack on IE unforgeability. However, we cannot make this reduction just yet, because the game creates more (cert, v) pairs that just the ones in set CCset^{gpk} . Therefore, we must first prune the game from all (cert, v) pairs s.t. $cert \notin \mathsf{CCset}^{gpk}$. We will do so in the next few games, by changing real instances $\mathsf{P}_{\mathsf{ssid}}$ so that if it runs on $cert \notin \mathsf{CCset}^{gpk}$ then it runs TRec on (π, td) in place of Rec on $(\pi, (gpk, bc), v)$.

GAME 3 (replacing Rec with TRec): We first modify Game 2 so that at initialization $\pi \leftarrow \mathsf{GPG}(1^{\kappa})$ is replaced by $(\pi, td) \leftarrow \mathsf{TGPG}(1^{\kappa})$. Secondly, we modify it so that if real session $\mathsf{P}_{\mathsf{ssid}}$ runs on a noncompromised certificate cert $\notin \mathsf{CCset}^{gpk}$, then $\mathsf{P}_{\mathsf{ssid}}$ sends $bc \leftarrow \mathsf{CertBlind}(cert)$ and runs $\mathsf{Snd}(\pi, (gpk, bc'))$ as before, but it replaces $\mathsf{Rec}(\pi, (gpk, bc), v)$ with $\mathsf{TRec}(\pi, (gpk, bc), td)$. CKEM setup indistinguishability and CKEM zero-knowledge imply that $|\operatorname{Pr}[\mathsf{G2}] - \operatorname{Pr}[\mathsf{G3}]| \leq \mathsf{negl}(\kappa)$.

GAME 4 (random key with non-compromised credentials): We modify Game 3 so that it acts like the ideal-world game in clauses 3b and 3(c)iii, i.e. $\mathsf{P}_{\mathsf{ssid}}$ outputs a random key if Link does not link bc' to any rt' in $\mathsf{RTset} \cup \mathsf{CCset}^{gpk}$ (see the discussion above). Let E be the event that some session $\mathsf{P}_{\mathsf{ssid}}$ receives bc'which is not linked to any $rt' \in \mathsf{CCset}^{gpk}$ (assume w.l.o.g. that RTset is empty) but \mathcal{A} queries H on K_S computed by $\mathsf{Snd}(\pi, (qpk, bc'))$ on this session. Assume that $\Pr[E]$ is non-negligible and we will argue that this violates our assumptions. Observe that if $\Pr[E]$ is non-negligible then so is an advantage of a tester \mathcal{T} which emulates game 3 by interacting with IE group manager for group gpk as a black-box, so whenever there is a certificate compromise query CompCert for this group, \mathcal{T} gets a new (cert, v) pair from the manager, and this certificate is placed in list CCset^{gpk} . However, \mathcal{T} also gets other certificates cert and revocation tokens rt, corresponding to non-compromised certificates, and on these it emulates game 3 using $\mathsf{TRec}(\pi, \cdot, td)$ as an oracle. When \mathcal{A} sends bc' to a chosen instance $\mathsf{P}_{\mathsf{ssid}}$ (the argument considers a separate reduction for each protocol instance invoked by \mathcal{Z}), \mathcal{T} sets x = (gpk, bc') and after interacting with challenge oracle $\operatorname{Snd}(\pi, x)$ it can distinguish between K_S output by $\operatorname{Snd}(\pi, (gpk, bc'))$ and a random key: \mathcal{T} simply monitors the queries of \mathcal{A} to H to decide if K_S is real or random. (Note that \mathcal{A} can send bc' after seeing Snd's messages, but we assume Snd can be split into Snd.1, which does not statement x as an input and produces all network messages, and Snd.2 which takes x and completes Snd execution to compute its local output K_S . Therefore \mathcal{T} can ask oracle Snd first for its network messages, which it forwards to \mathcal{A} , and only then gives statement x to this oracle.) By CKEM strong simulation-soundness, if \mathcal{T} 's advantage in this distinguishing game is non-negligible, then there exists an efficient extractor which interacts with the same group manager and outputs with non-negligible probability witness v for (gpk, bc') in relation $\mathcal{R}^{\mathsf{IE}}$, i.e. s.t. $\mathsf{Ver}(gpk, bc, v) = 1$. However, since event E assumes that Link(gpk, bc', rt) = 0 for all rt's corresponding to the compromised certificates, an existence of such extractor is contradicted by IE unforgeability. This implies that $\Pr[E]$ is negligible, which in turn implies that $|\Pr[G3] - \Pr[G4]| \le \operatorname{negl}(\kappa)$.

Game reassessment: By the changes made above, Game 4 acts like the ideal-world in all of step 3. However, in step 4 it acts close to the ideal-world game except for three aspects: First, it simulates instances P_{ssid} running on all non-compromised credentials, while the ideal-world game does so only on hidden credentials (recall that if credential is hidden then it is not compromised, but not the other way around because credentials can have revealed revocation tokens without being compromised). Second, it simulates such instances using TRec instead of psTRec. Third, on simulated selections it creates bc via CertBlind(*cert*) rather than sampling bc at random. Fourth, Game 4 in step 4c checks only if there exists $rt \in \mathsf{RTset}^{gpk}$ s.t. Link(gpk, bc', rt) = 1, in contrast to the ideal-world procedure which casts a wider net and checks if there exists gpk^* and $rt \in \mathsf{RTset}^{gpk^*}$ s.t. Link(gpk^* , bc', rt) = 1. We address each of these divergence in turn in the following games.

GAME 5 (running instead of simulating sessions on revealed-only credentials): We modify Game 4 by restricting the set of parties emulated via TRec to those who execute on *cert* which is "hidden", i.e. they were not subject to either CompCert or RevealRT queries (i.e. *cert* which corresponds to *rt* s.t. $rt \notin \mathsf{RTset}^{gpk}$), instead of parties who run on *cert* which is only non-compromised, i.e. *cert* $\notin \mathsf{CCset}^{gpk}$. The change is that we change TRec back to Rec on these instances, hence *CKEM zero-knowledge* implies that $|\Pr[\mathsf{G4}] - \Pr[\mathsf{G5}]| \leq \mathsf{negl}(\kappa)$.

GAME 6 (replacing TRec with psTRec): We modify Game 5 to eliminate the second divergence listed above, i.e. (1) at initialization TGPG(1^{κ}) is replaced by psTGPG(1^{κ}), and (2) all sessions P_{ssid} which run on hidden certificates (i.e. certificates whose revocation tokens rt satisfy $rt \notin \text{RTset}^{gpk}$) are emulated using simulator psTRec(π , \cdot , td) for td output by psTGPG, instead of TRec(π , \cdot , td) for td output by TGPG, as was done in Game 4. The *CKEM setup indistinguability*, *CKEM zero-knowledge*, and *CKEM statement-postponed zero-knowledge* properties together imply that $|\Pr[G5] - \Pr[G6]| \leq \operatorname{negl}(\kappa)$.

GAME 7 (random certs for hidden certificates): We modify Game 6 to eliminate the third divergence, i.e. if P_{ssid} runs on a hidden certificate cert then we ignore cert and we sample bc at random from the IE certificate space D. The difference this creates is that on hidden sessions corresponding to any fixed certificate *cert*, each *bc* is independently sampled from D whereas before they were all created via CertBlind(*cert*). However, *IE covertness* implies that these two views are indistinguishable (note that in the IE covertness game the tester gets the IE master secret key *msk* so it can create other valid certificate, witness pairs and thus emulate the rest of the game). Consequently, $|\Pr[G6] - \Pr[G7]| \leq \operatorname{negl}(\kappa)$.

GAME 8 (general revocation search): We modify Game 6 so clause 4c casts a "wider net" by checking for any gpk^* , rt s.t. Link(gpk^* , bc', rt) = 1 and $rt \in \mathsf{RTset}^{gpk^*}$. Note that if $gpk^* = gpk$ then this is identical to the previous constraint. Furthermore, if this clause is satisfied for $gpk^* \neq gpk$ but not with gpkthen $\mathsf{P}_{\mathsf{ssid}}$ outputs random key both before and after this change. The only externally observable difference this change creates in the game is in case the clause is satisfied for $both \, gpk^*$ and $gpk \neq gpk^*$. However, *IE unambiguity* implies that such even has negligible probability, hence $|\operatorname{Pr}[\mathsf{G7}] - \operatorname{Pr}[\mathsf{G8}]| \leq \operatorname{negl}(\kappa)$.

GAME 9 (simulated random beacons): We modify Game 8 so that when \mathcal{Z} tells $\mathsf{P}_{\mathsf{ssid}}$ to run $\mathsf{Auth}^{\$(\kappa)}$, we replace $\mathsf{P}_{\mathsf{ssid}}$ executing the random beacon $\mathsf{Auth}^{\$(\kappa)}$ with the corresponding view in the ideal-world, i.e. we run substeps (s1)-(s3) of the ideal-world execution in step 2, i.e. we send random bc sampled from the IE certificate space, replace $\mathsf{Snd}^{\$(\kappa)}$ with $\mathsf{Snd.1}(\pi)$ on the outgoing CKEM, and replace $\mathsf{Rec}^{\$(\kappa)}$ with $\mathsf{psTRec}(\pi, td)$ on the incoming CKEM, but $\mathsf{P}_{\mathsf{ssid}}$ output is still \bot . Note that after this change the execution emulates the ideal-world in step 2. The first change creates an indistinguishable statistical distance because by covertness property of IE the domain of blinded certificates is uniformly encodeable. The second change is computationally indistinguishable by sender covertness of CKEM, and the third change is computationally indistinguishable by receiver covertness of CKEM, which says that $\mathsf{Rec}^{\$(\kappa)}$ can be replaced by Rec on an arbitrary statement,witness pair (x, w) in the language relation, and statement-postponed zero-knowledge of CKEM which shows that such Rec is indistinguishable from $\mathsf{psTRec}(\pi, td, x)$. However, since the adversary only sees network messages here, i.e. it has access to Rec and not $\mathsf{Rec}_{\&Out}$, we can restrict the last transition to $\mathsf{psTRec}.1(\pi, td)$, as required in Game 9. In particular, the arbitrary (x, w) in the language relation is relevant only to the mid-point in the above transition, i.e. to show that $\mathsf{Rec}^{\$(\kappa)} \approx_c \mathsf{psTRec}.1(\pi, td)$ by the two transitions, $\mathsf{Rec}^{\$(\kappa)} \approx_c \mathsf{Rec}(\pi, x, w)$ and $\mathsf{Rec}(\pi, x, w) \approx_c \mathsf{psTRec}(\pi, td)$. Summing up, the above four properties imply that $|\mathsf{Pr}[\mathsf{G8}] - \mathsf{Pr}[\mathsf{G9}]| \leq \mathsf{negl}(\kappa)$.

GAME 10 (*ideal world*): This is the ideal-world game, but by inspection it is identical to Game 9, hence Pr[G9] = Pr[G10], which completes the proof.

C Proof of Concept Implementation

Our proof of concept implementation can be downloaded at the following link¹⁸ and a video of a CKEM demo can be found at this link¹⁹. We emphasize that our covert encodings on BLS12-381 in sage is a proof of concept, i.e., it is somewhat slow. However, moving the implementation to C++ (NTL) would significantly improve the timings. We leave this to future work due to deadline constraints.

Here we describe an implementation written in sage and C++ using both the MIRACL library as well as Shoup's Number Theory Library²⁰ (NTL). We emphasize that though there are many implementations of hashing onto a curve, there are *few* implementations of bandwidth-efficient, advanced covert protocols. Standard covert encodings we use are described in Section 2. The commitment scheme and the SPHF are instantiated on Curve25519 [11]. Further, we used the BLS12-381 pairing for our pairing friendly type-3 curve [14]. This curve is implemented in MIRACL.

To our knowledge, there are no implementations of covert encodings of type-3 curves. However, the required building blocks for our protocol are implemented and described in the literature. The first main building block is Elligator Squared [59], which works on all elliptic curves. The second is the work of Wahby and Boneh [62] showing how to apply standard encodings to BLS12-381.

Elligator Squared requires an encoding $f : \mathbb{F}_q \to E(\mathbb{F}_q)$ which is well-distributed and has few preimages for each point in its range. Then, a point on the curve is encoded by two base field elements u, v randomly sampled conditioned on f(u) + f(v) = P. (See Algorithm 1 of [59] for more details.)

¹⁸ https://www.dropbox.com/sh/dvxg0e0xuw3qtlj/AAAO1CKF-OWAD1QANzf5KaqFa?dl=0

¹⁹ https://www.dropbox.com/s/cnpbkgf55n2hyyu/short_cAKE_demo-Copy.mp4?dl=0

²⁰ https://www.sagemath.org/, https://github.com/miracl/core/tree/master/cpp, and https://libntl. org/

Unfortunately, the only encoding directly applicable to BLS12-381 is the slow, complicated encoding given by Shallue and van de Woestijne [56]. Furthermore, there is no simple description of the inverse map in the literature to our knowledge. However, Wahby and Boneh give a simple solution in their work on hashing to BLS12-381. They simply compute a low degree isogeny , $\phi : \mathbb{G}_1 \to \mathbb{G}'_1$ which is also invertible, to an alternative curve where the simplified SWU map given by Brier et al. [15] applies! Therefore, we perform the following to covertly encode a point P on BLS12-381's \mathbb{G}_1 :

- 1. First, map the point using the isogeny $P' = \phi(P)$ given by Wahby and Boneh.
- 2. Then, use Elligator Squared with the simplified SWU encoding, as described in Section 4 of [59].

Bandwidth Comparison. Here we compare the required bandwidth between our protocol and the previous state of the art [39]. In particular, we measure the number of bytes a party sends in its message in the Auth protocol. Our protocol has a message of $(\sigma, w, c_1, c_2, z, hp)$ which is two elements in \mathbb{G}_1 , three elements in \mathbb{G} , and one exponent for \mathbb{G}_1 . Therefore, we have $(4 \cdot 381 + 128)/8 \approx 207$ bytes for the two \mathbb{G}_1 elements, $(255+128) \approx 48$ for the exponent z, and $2 \cdot (256+128)/8 = 96$ bytes for the two \mathbb{G} , Curve25519, elements encoded with Elligator2 together with rejection sampling. Summed, the total bandwidth is 351 bytes! This is significantly less than the 3.6KB in [39], by a factor of ≈ 10 . Further, we estimate comparable security in both schemes²¹.

²¹ NIST estimates RSA with 2048 bit modulus has about 112 bits of security, https://nvlpubs.nist.gov/ nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf, and NCC recently estimated BLS12-381 as having 117 bits of security, https://research.nccgroup.com/wp-content/uploads/2020/07/NCC_Group_Zcash2018_ Public_Report_2019-01-30_v1.3.pdf.