Short Concurrent Covert Authenticated Key Exchange (Short cAKE)

Karim Eldefrawy[1], Nicholas Genise[2], and Stanislaw Jarecki[3](✉)

[1] SRI International, Menlo Park, USA
karim.eldefrawy@sri.com
[2] Duality Technologies, Hoboken, USA
ngenise@dualitytech.com
[3] University of California, Irvine, Irvine, USA
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.[1] 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

[1] The full version of this paper appears in [22].

N. Genise—This work was done while the second author was at SRI International.

J. Guo and R. Steinfeld (Eds.): ASIACRYPT 2023, LNCS 14445, pp. 75–109, 2023.
https://doi.org/10.1007/978-981-99-8742-9_3

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. [31] for the two-party case, and by Chandran et al. [15] and Goyal and Jain [29] for the multi-party case. Both [15,29,31] prove feasibility for covert computation of arbitrary circuits which tolerates passive and malicious adversaries, respectively. Subsequently, Jarecki [33] 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 [32], 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 [32] achieves only sequential security, and does not ensure independence of keys across sessions, which is insufficient for full-fledged (covert) AKE.[2] Regarding round complexity and bandwidth, the cAKE protocol in [32] requires 6 message flows and relies on a composite-order group (and a factoring assumption), resulting in bandwidth which can be estimated as at least 3.6 kB. Recent works on random encodings of elliptic curve points, e.g. [8,47], 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 $\mathcal{F}_{\text{cAKE}}[\mathsf{C}]$ shown in Fig. 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 $\mathcal{F}_{\text{AKE}}[\mathsf{C}, \mathsf{L}]$, characterized by C and a *leakage function* L, which is portrayed in the same figure. Reading Fig. 1 with dashed text and without greyed text defines $\mathcal{F}_{\text{AKE}}[\mathsf{C}, \mathsf{L}]$, and with greyed text and without dashed text defines $\mathcal{F}_{\text{cAKE}}$.

In an AKE protocol, i.e. a protocol that realizes $\mathcal{F}_{\text{AKE}}$, 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

[2] In particular, [32] does not imply security against man in the middle attacks.

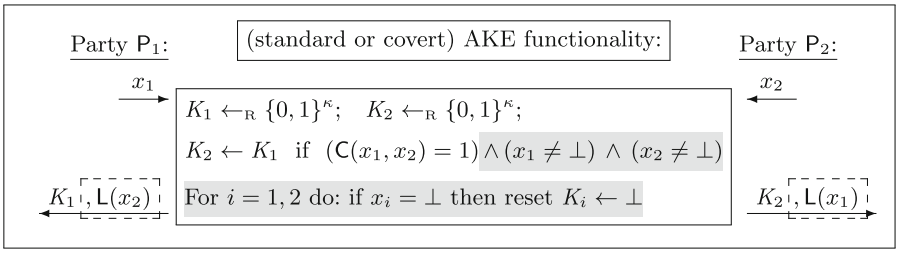

Fig. 1. Standard AKE functionality $\mathcal{F}_{\text{AKE}}[\mathsf{C}, \mathsf{L}]$ includes dashed text & omits greyed text; Covert AKE functionality $\mathcal{F}_{\text{cAKE}}[\mathsf{C}]$ includes greyed text & omits dashed text.

key $K_1 = K_2$, otherwise their outputs K_1, K_2 are independent.[3] If L is a non-trivial function, then the protocol leaks $\mathsf{L}(x)$ on P's input x to P's counterparty.

For example, Password Authenticated Key Exchange (PAKE) [5] can be defined as (secure realization of) $\mathcal{F}_{\text{AKE}}[\mathsf{C}_{\mathsf{pa}}]$ where C_{pa} is an equality test, i.e., $\mathsf{C}_{\mathsf{pa}}(x_1, x_2) = 1$ if and only if $x_1 = x_2$. In another example, a standard notion of AKE, e.g. [21], which we will call here as a *Fixed Public Key* AKE (FPK-AKE) to distinguish it from other AKE types, can be defined as $\mathcal{F}_{\text{AKE}}[\mathsf{C}_{\mathsf{fpk}}, \mathsf{L}_{\mathsf{fpk}}]$ where $\mathsf{C}_{\mathsf{fpk}}(x_1, x_2) = 1$ 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 $\mathsf{L}_{\mathsf{fpk}}$ is typically omitted in the works on FPK-AKE, e.g. [3,14], 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 $\mathsf{L}_{\mathsf{fpk}}((sk_{\mathsf{P}}, pk_{\mathsf{CP}})) = (pk_{\mathsf{P}}, pk_{\mathsf{CP}})$ where pk_{P} is a public key corresponding to sk_{P}.[4]

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

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 mali-

[3] Note that Fig. 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.

[4] In a standard FPK-AKE protocol party P can reveal either key. E.g. Sigma [36] used in TLS reveals P's own key pk_{P}, while SKEME [35] reveals key pk_{CP} which party P assumes for its counterparty, unless it employs key-private encryption [4].

cious P_2^* cannot distinguish an interaction with a real party P_1 (where $x_1 \neq \perp$) and a random beacon (where $x_1 = \perp$) because in either case $\mathcal{F}_{\text{cAKE}}$ gives P_2^* the same output, a random key K_2. Indeed, the only way P_2^* can distinguish cAKE participant 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 $\mathcal{F}_{\text{cAKE}}$:

(1) P_1 = protocol party with x_1 s.t. $\mathsf{C}(x_1, x_2^*) = 1$, 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 $x_1 = \perp$, in which case $K_1 = \perp$.

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.[5] 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 $\mathsf{C}_\mathsf{G}(x_1, x_2)$ outputs 1 if and only if $\exists\, gpk$ s.t. $x_1 = (gpk, cert_1)$, $x_2 = (gpk, cert_2)$, and $\mathsf{Ver}(gpk, cert_1) = \mathsf{Ver}(gpk, cert_2) = 1$, where Ver stands for certificate verification. In other words, both parties must assume the same group identified by gpk and 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. $\mathsf{Ver}(gpk, cert^*) = 1$ and $\forall\, i\ cert^* \neq cert_i$.

The above setting of group cAKE is the same as that of *group signatures* [16], except that membership certificates are used to authenticate, not to sign,[6] 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 $\mathcal{F}_{\text{AKE}}[\mathsf{C}_\mathsf{G}, \mathsf{L}]$ where leakage L

[5] This requires encryption with ciphertexts indistinguishable from random bitstrings, but this is achieved by standard block cipher modes, CBC, OFB, or RND-CTR.
[6] Using group signatures for authentication is known as an *Identity Escrow* [34].

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.[7]

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 $\mathsf{Ver}(gpk, cert) = 1$ then $\mathsf{Link}(cert, rt) = 1$ for *rt* associated with *cert*, and (2) certificates remain unforgeable in the presence of revocation tokens.[8] If $\mathsf{Link}(cert, \mathsf{RTset})$ stands for a procedure which outputs 1 iff $\exists\, rt \in \mathsf{RTset}$ s.t. $\mathsf{Link}(cert, rt) = 1$, then we define *group covert AKE (with revocation)*, or simply *group cAKE*, as $\mathcal{F}_{\mathrm{cAKE}}[\mathsf{C}_{\mathsf{Gwr}}]$ where $\mathsf{C}_{\mathsf{Gwr}}(x_1, x_2) = 1$ iff

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

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 [44] or targeting it for filtering or other attacks [46,48]. *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 [11]) 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 communication for Everyone (RACE) [45]. The RACE program objective was to develop "an 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 [45]. An efficient covert authentication could be play an essential role in such a system.

[7] *Secret Handshake* [2] flips this leakage, realizing $\mathcal{F}_{\mathrm{AKE}}[\mathsf{C}_{\mathsf{G}}, \mathsf{L}']$ for L' that hides *gpk* but reveals a one-way function of P_i's certificate. To complete comparisons, standard PKI-based AKE realizes $\mathcal{F}_{\mathrm{AKE}}[\mathsf{C}_{\mathsf{G}}, \mathsf{L}'']$ 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.

[8] Here we follow the *verifier-local revocation* model [10], but other models are possible, e.g. using cryptographic *accumulators* [6,12].

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 $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{pa}}]$, for C_{pa} defined above. Several known efficient PAKE schemes, e.g. EKE [5] and SPAKE2 [1], most likely realize $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{pa}}]$ 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 fixed-length bitstring, see Sect. 2.1. (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 $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{fpk}}]$, for $\mathsf{C}_{\mathsf{fpk}}$ defined above. The work on *key-hiding AKE* [30] shows that several FPK-AKE protocols, namely 3DH [40], HMQV [37], and SKEME [35] instantiated with key-private and PCA-secure encryption, realize $\mathcal{F}_{\text{AKE}}[\mathsf{C}_{\mathsf{fpk}}]$, i.e. FPK-AKE without leakage, and after similar implementation adjustments as in the case of SPAKE2, these protocols probably realize $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{fpk}}]$. (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 encodable ciphertexts, such as the Boneh-Franklin IBE [9] 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 [17,39] 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. [23], 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) [13] 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) [15,32],[9] 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 [20]. Covert CKEM is a covert KEM version of Witness Encryption [26]: 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 [32]. Namely, it constructs group cAKE generically from a covert Identity Escrow (IE) scheme [34] 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 $(cert \in \mathcal{L}^{\mathsf{IE}}(gpk')) \wedge (cert' \in \mathcal{L}^{\mathsf{IE}}(gpk))$ 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.[10] 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 [32] (for the same general class of languages with Sigma-protocols):
(1) First, we combine strong soundness of [32] and simulation-soundness of [7] 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

[9] Covert CKEM was called *ZKSend* in [15]. Variants of (covert or non-covert) CKEM notion include *Conditional OT* [19], *Witness Encryption* [26], and *Implicit ZK* [7].

[10] 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.

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. [33]. 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 $\mathsf{S}(x)$ instance, even if $\mathcal{A}$ has access to (simulated) $\mathsf{R}(x')$ 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 [32]. First, we rely on smooth projective hash functions (SPHF) with a property akin to PCA (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 [42]. The resulting IE certificates involve only two elements

of a type-3 bilinear pairing curve [25], which can be covertly encoded using the Elligator Squared encoding of Tibouchi [47], with a hash onto group due to Wahby and Boneh [50]. 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. [8].

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. [41]) 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 [49] 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. [15] 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 [29] 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 [32] 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 [38] gave the first post-quantum covert group-based AKE with trusted setup by adopting Jarecki's construction [32] 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 [27].

None of the aforementioned works are proven secure in the UC framework [13]. Cho, Dachman-Soled, and Jarecki [17] achieve UC security for covert MPC of two specific functionalities, namely string equality and set intersection. The work of Jarecki [33] 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 [32] is secure as-is in the UC model despite [32] 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 Sect. 6. For space constraint reasons, all security proofs, and an overview of our proof of concept implementation, are deferred to the full version of the paper [22].

2 Preliminaries

We reserve κ for the security parameter throughout the paper. The uniform distribution on a finite set S is denoted as $\mathcal{U}(S)$. We write $x \leftarrow_{\mathrm{R}} \mathcal{X}$ for a random variable sampled from distribution $\mathcal{X}$, and we write $x \leftarrow_{\mathrm{R}} S$ for $x \leftarrow_{\mathrm{R}} \mathcal{U}(S)$.

Standard Notation, Σ-Protocols. For lack of space, we defer the review of standard notions of computational and statistical indistinguishability, notation for groups with bilinear maps, and the review of Σ-protocols, a special form of honest-verifier zero-knowledge proof of knowledge [20], to the full version of the paper on eprint [22]. We note that in this work we assume a slightly *strengthened form of Σ-protocols* than in [20], where (1) both the verifier and the simulator use the same function to recompute the prover's first message from the rest of the transcript, (2) prover's response is a deterministic function of prior messages, and (3) the simulator samples that response from some uniformly encodable domain (see [22] for more details).

2.1 Covert Encodings and Random Beacons

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

Definition 2.1. *Functions* $(\mathsf{EC}, \mathsf{DC})$ *form a* covert encoding of domain D *if there is an l s.t.* $\mathsf{EC} : D \rightarrow \{0,1\}^l$, $\mathsf{DC} : \{0,1\}^l \rightarrow D$ *is an inverse of* EC, *and* $\mathsf{EC}(\mathcal{U}(D))$ *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* $\mathsf{EC}(\mathcal{U}(D); r)$ *to be statistically close to uniform when* EC*'s randomness r is a uniform random bitstring of fixed length.*

Definition 2.2. *We call a finite set S* uniformly encodable *if it has a covert encoding. Further, a family of sets $\mathcal{S} := \{S[\pi]\}_{\pi \in \mathcal{I}}$ 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 $r \leftarrow_{\mathrm{R}} [m]$ where $m = \lceil 2^{\log_2(n)+\kappa}/n \rceil$. (See e.g. Sect. 3.4 of [47] for a proof.) For uniform encodings of elliptic curve points we require two sub-cases: (2a) a curve in Montgomery form and (2b) a pairing friendly curve. In case (2a) we can use the Elligator-2 encoding [8], 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. [50], a simple way to encode point P sampled from *the whole group*, i.e. $P \leftarrow_{\mathrm{R}} \mathcal{U}(\mathbb{G})$ as opposed to $P \leftarrow_{\mathrm{R}} \mathcal{U}(S)$, is to sample $r \leftarrow_{\mathrm{R}} \{0,1\}^\kappa$ until $Q = \mathsf{H}(r) + P$ is in S, where G is a generator of $\mathbb{G}$, and output $z = \text{Elligator-2}(Q)||r$ (see [22]). In case (2b) we can use Tibouchi's Elligator Squared encoding [47], 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.

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 $\mathsf{B}^{\$(\kappa)}$ 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 $\mathsf{B}^{\$(\kappa)}$ 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 steganographic channel [31] 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 $(\mathsf{KG}, \mathsf{CG}, \mathsf{Auth})$ with the following input/output behavior:

- KG is a key generation algorithm, used by the group manager, s.t. $\mathsf{KG}(1^\kappa)$ 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. $\mathsf{CG}(msk)$ 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 $K \in \{0,1\}^\kappa \cup \{\perp\}$ is a session key (or $\perp$ if no key is established) and $rt \in \mathsf{RTset} \cup \{\perp\}$ is a detected revocation token in RTset, or $\perp$ 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 $K \neq \perp$ 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 $K = \perp$, which violates the (simplified) covert mutual authentication model of Fig. 1 in Sect. 1. However, w.l.o.g. P is free to run any upper-layer protocol Π that utilizes Auth output K by replacing $K = \perp$ 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 $\mathcal{F}_{\text{g-cAKE}}$ shown in Fig. 2, and we say that scheme $\Pi = (\mathsf{KG}, \mathsf{CG}, \mathsf{Auth})$ is a group cAKE if Π UC-realizes functionality $\mathcal{F}_{\text{g-cAKE}}$ in the standard sense of universal composability [13]. However, we adapt the UC framework [13] to the covert computation setting so that environment $\mathcal{Z}$ can pass to party P executing an AKE protocol Auth a special input $\perp$, which causes party P to play a role of a random beacon. (The same convention was adopted by Chandran et al. [15] 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 $(\mathsf{NewSession}, \mathsf{ssid}, \perp)$ from $\mathcal{Z}$, this party's session indexed by identifier ssid is replaced by a random beacon, i.e., it will run $\mathsf{Auth}^{\$(\kappa)}$ instead of Auth, see Sect. 2.

In Definition 3.1 we use the notation of [13], where $\mathbf{Ideal}_{\mathcal{F}_{\text{g-cAKE}}, \mathcal{A}^*, \mathcal{Z}}(\kappa, z)$ 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 $\mathcal{F}_{\text{g-cAKE}}$, 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 $(\mathsf{NewSession}, \mathsf{ssid}, \perp)$.

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

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

Group cAKE Functionality. We explain how functionality $\mathcal{F}_{\text{g-cAKE}}$ operates and how it differs from a standard AKE functionality, e.g. [14,37]. Note that functionality $\mathcal{F}_{\text{g-cAKE}}$ in Fig. 2 is much more complex than functionality $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{Gwr}}]$ in Fig. 1 in Sect. 1. The first difference are environment commands GInit and $\mathsf{CertInit}$, which are used to initialize groups and generate membership certificates, and commands $\mathsf{CompCert}$ and $\mathsf{RevealRT}$, which model adversarial compromise of resp. certificates and revocation tokens (which are not assumed public by default). Command $\mathsf{NewSession}$ models party P engaging in group cAKE on input $x = (gpk, cert, \mathsf{RTset})$, exactly as $\mathcal{F}_{\text{cAKE}}[\mathsf{C}_{\mathsf{Gwr}}]$ of Fig. 1, except that in $\mathcal{F}_{\text{g-cAKE}}$ these real-world inputs are replaced by ideal-world identifiers, resp. $\mathsf{gid}, \mathsf{cid}, \mathsf{RTcids}$. One aspect of functionality $\mathcal{F}_{\text{g-cAKE}}$ 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 $\mathcal{F}_{\text{g-cAKE}}$ 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.

Below we explain how we model secure initialization and party interactions with the group manager, and we briefly overview how we model compromise of credentials and revealing of revocation tokens, and how $\mathcal{F}_{\text{g-cAKE}}$ models key establishment and active (or passive) session attacks. For a more detailed walk through functionality $\mathcal{F}_{\text{g-cAKE}}$, see the eprint version of this paper [22].

Secure Initialization and Trusted Group Manager. A crucial difference between $\mathcal{F}_{\text{g-cAKE}}$ 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 [30], maybe accessing a global certificate functionality, e.g. as in [14]. By contrast, the Covert AKE model $\mathcal{F}_{\text{g-cAKE}}$ must explicitly include a *group manager* party, denoted GM, initialized via query $(\mathsf{GInit}, \mathsf{gid})$ which models generation of a group public key indexed by a unique identifier gid. Consequently, the $\mathcal{F}_{\text{g-cAKE}}$ 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 $(\mathsf{GInit}, \mathsf{gid})$ 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

$\mathcal{F}_{\text{g-cAKE}}$ interacts with parties denoted P and GM and adversary $\mathcal{A}^*$. Sets $\mathsf{CompCert}^{\mathsf{gid}}$ and $\mathsf{RevRT}^{\mathsf{gid}}$ store resp. revealed certificates and revocation tokens for each gid.

Keys: Initialization and Attacks

On $(\underline{\mathsf{GInit}}, \mathsf{gid})$ from GM:
Save $(\mathsf{gid}, \mathsf{GM})$, reject future GInit queries for the same gid, send $(\mathsf{GInit}, \mathsf{GM}, \mathsf{gid})$ to $\mathcal{A}^*$.

On $(\underline{\mathsf{CertInit}}, \mathsf{gid}, \mathsf{cid})$ from P:
If $\exists$ no prior record $(\cdot, \mathsf{gid}, \mathsf{cid})$, save tuple $(\mathsf{P}, \mathsf{gid}, \mathsf{cid})$.

On $(\underline{\mathsf{CompCert}}, \mathsf{P}, \mathsf{gid}, \mathsf{cid})$ from $\mathcal{A}^*$ [$\mathcal{A}^*$ needs environment permission for this action]:
If $\exists$ rec. $(\mathsf{P}, \mathsf{gid}, \mathsf{cid})$ and $\exists$ no rec. $(\mathsf{P}, \cdot, \mathsf{gid}, \mathsf{cid}, \cdot, \cdot)$ add cid to $\mathsf{CompCert}^{\mathsf{gid}}$ and $\mathsf{RevRT}^{\mathsf{gid}}$.

On $(\underline{\mathsf{RevealRT}}, \mathsf{P}, \mathsf{gid}, \mathsf{cid})$ from $\mathcal{A}^*$ [$\mathcal{A}^*$ needs environment permission for this action]:
If $\exists$ record $(\mathsf{P}, \mathsf{gid}, \mathsf{cid})$ and $\exists$ no record $(\mathsf{P}, \cdot, \mathsf{gid}, \mathsf{cid}, \cdot, \cdot)$ add cid to $\mathsf{RevRT}^{\mathsf{gid}}$.

Authentication Sessions: Initialization, Connections, Attacks

On $(\underline{\mathsf{NewSession}}, \mathsf{ssid}, \perp)$ from P:
Save record $(\mathsf{P}, \mathsf{ssid}, \perp, \perp, \perp, \perp)$ marked random, send $(\mathsf{NewSession}, \mathsf{P}, \mathsf{ssid}, \perp)$ to $\mathcal{A}^*$.

On $(\underline{\mathsf{NewSession}}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids})$ from P:
If $\mathsf{RTcids} \subseteq \mathsf{RevRT}^{\mathsf{gid}}$ and $\exists$ record $(\mathsf{P}, \mathsf{gid}, \mathsf{cid})$ but $\exists$ no prior record $(\mathsf{P}, \mathsf{ssid}, \cdot, \cdot, \cdot, \cdot)$:
- if $\mathsf{cid} \notin \mathsf{RevRT}^{\mathsf{gid}}$, send $(\mathsf{NewSession}, \mathsf{P}, \mathsf{ssid}, \perp)$ to $\mathcal{A}^*$
- if $\mathsf{cid} \in \mathsf{RevRT}^{\mathsf{gid}}$, send $(\mathsf{NewSession}, \mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid})$ to $\mathcal{A}^*$

Save record $(\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids}, \perp)$ marked fresh.

On $(\underline{\mathsf{Interfere}}, \mathsf{P}, \mathsf{ssid})$ from $\mathcal{A}^*$:
If $\exists$ record $(\mathsf{P}, \mathsf{ssid}, \cdot, \cdot, \cdot, \perp)$ marked fresh, re-label it $\mathsf{interfered}$.

On $(\underline{\mathsf{Connect}}, \mathsf{P}, \mathsf{ssid}, \mathsf{P}', \mathsf{ssid}')$ from $\mathcal{A}^*$:
If $\exists$ record $\mathsf{rec} = (\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \cdot, \perp)$ marked fresh and record $(\mathsf{P}', \mathsf{ssid}', \mathsf{gid}', \mathsf{cid}', \cdot, K')$ marked either fresh or $\mathsf{connected}(\mathsf{P}, \mathsf{ssid}, \mathsf{cid})$ (any of gid', cid', K' can equal $\perp$) then:
- if $\mathsf{gid} = \mathsf{gid}'$ then re-label rec as $\mathsf{connected}(\mathsf{P}', \mathsf{ssid}', \mathsf{cid}')$
- if $\mathsf{gid} \neq \mathsf{gid}'$ then re-label rec as $\mathsf{interfered}$

On $(\underline{\mathsf{Impersonate}}, \mathsf{P}, \mathsf{ssid}, \mathsf{gid}^*, \mathsf{cid}^*)$ from $\mathcal{A}^*$:
If $\exists$ $\mathsf{rec} = (\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \cdot, \cdot, \perp)$ marked fresh:
- if $\mathsf{gid} = \mathsf{gid}^*$ and $\mathsf{cid}^* \in \mathsf{CompCert}^{\mathsf{gid}}$ then re-label rec as $\mathsf{compromised}(\mathsf{cid}^*)$
- if $\mathsf{gid} = \mathsf{gid}^*$ and $\mathsf{cid}^* \notin \mathsf{CompCert}^{\mathsf{gid}}$ then re-label rec as $\mathsf{interfered}(\mathsf{cid}^*)$
- if $\mathsf{gid} \neq \mathsf{gid}^*$ then re-label rec as $\mathsf{interfered}$

Authentication Sessions: Key Establishment

On $(\underline{\mathsf{NewKey}}, \mathsf{P}, \mathsf{ssid}, K^*)$ from $\mathcal{A}^*$:
If $\exists$ session record $\mathsf{rec} = (\mathsf{P}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids}, \perp)$ marked flag then:
1. if $\mathsf{flag} = \mathsf{random}$ set $(K, \mathsf{cid}_{\mathsf{CP}}) \leftarrow (\perp, \perp)$
2. if $\mathsf{flag} = \mathsf{compromised}(\mathsf{cid}')$ for $\mathsf{cid}' \notin \mathsf{RTcids}$, set $(K, \mathsf{cid}_{\mathsf{CP}}) \leftarrow (K^*, \perp)$
3. if flag is either $\mathsf{connected}(\cdot, \cdot, \mathsf{cid}')$ or $\mathsf{compromised}(\mathsf{cid}')$ or $\mathsf{interfered}(\mathsf{cid}')$, for $\mathsf{cid}' \in \mathsf{RTcids}$, set $(K, \mathsf{cid}_{\mathsf{CP}}) \leftarrow (\perp, \mathsf{cid}')$
4. if $\mathsf{flag} = \mathsf{connected}(\mathsf{P}', \mathsf{ssid}', \mathsf{cid}')$, $\mathsf{cid}' \notin \mathsf{RTcids}$, and $\exists$ $\mathsf{rec}' = (\mathsf{P}', \mathsf{ssid}', \mathsf{gid}, \mathsf{cid}', \cdot, K')$ s.t. $K' \neq \perp$ and rec' terminated as $\mathsf{connected}(\mathsf{P}, \mathsf{ssid}, \mathsf{cid})$, set $(K, \mathsf{cid}_{\mathsf{CP}}) \leftarrow (K', \perp)$
5. in any other case set $K \leftarrow_{\mathrm{R}} \{0,1\}^{\kappa}$ and $\mathsf{cid}_{\mathsf{CP}} \leftarrow \perp$

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

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 (CertInit, 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 CertInit 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.[11]

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 (RevealRT, gid, P, cid). The first command adds cid to the set $\mathsf{CompCert}^{\mathsf{gid}}$ of compromised certificate identifiers in group gid, and both commands add cid to the set $\mathsf{RevRT}^{\mathsf{gid}}$ 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

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

requiring that the adversary corrupts all certificates and reveals all revocation tokens at the beginning of the interaction.

AKE Session Establishment and Attacks. Party P starts an AKE session via command $(\mathsf{NewSession}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids})$. Values $\mathsf{gid}, \mathsf{cid}, \mathsf{RTcids}$ can either form an input to a real protocol party, or they can be $\perp$, in which case this command triggers an execution of a random beacon. Crucially, if cid is not in $\mathsf{RevRT}^{\mathsf{gid}}$, i.e. a party runs on a certificate whose revocation token is not revealed, then $\mathcal{A}^*$ gets the same view of the real-world protocol as its view of the random beacon, i.e. $\mathcal{A}^*$ gets $(\mathsf{NewSession}, \mathsf{P}, \mathsf{ssid}, \perp)$ in either case. Below we will use a word "session" for both real sessions and random beacons. The adversary can react to sessions in 3 ways: (1) it can interfere in them, using query $\mathsf{Interfere}$, which makes real sessions output random keys K on termination, modeled by query NewKey (random beacon sessions always output $K = \perp$, regardless of adversarial behavior towards them); (2) it can passively connect them to another session, using query $\mathsf{Connect}$, which will make the two sessions establish a shared key at termination *if* they assume same group gid and use certificates which are not on each other's revocation lists (otherwise they output independent random keys); or (3) it can actively attack P's session using a compromised certificate cid^* for some target group gid^*, as modeled by query $\mathsf{Impersonate}$: If gid^* matches the gid used by P then $\mathcal{F}_{\text{g-cAKE}}$ marks P's session $\mathsf{compromised}(\mathsf{cid}^*)$, but when this session terminates via NewKey then $\mathcal{F}_{\text{g-cAKE}}$ lets $\mathcal{A}^*$ set its key to K^* only if cid^* is not in RTcids used by P. Otherwise P outputs $K = \perp$ and cid^* as the identifier of a revoked party which P "caught" in this interaction.

(For a more detailed walk-through of the $\mathcal{F}_{\text{g-cAKE}}$ session attack and termination interfaces see the eprint version of the paper [22].)

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 bea-

cons. 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.

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 Fig. 6 in Sect. 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 Subsect. 4.1, PCA-secure covert SPHF, in Subsect. 4.2, and covert IE, in Subsect. 4.3. (The construction of covert CKEM using trapdoor commitments and PCA-secure SPHF is shown in Sect. 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 [24] 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* $(\mathsf{GPG}, \mathsf{PG}, \mathsf{Com}, \mathsf{Decom})$ *forms a* trapdoor commitment scheme *if there exists algorithms* $(\mathsf{TPG}, \mathsf{TCom}, \mathsf{TDecom})$ *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 decommimtment* 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* $m \in \mathcal{M}$,

$$\begin{aligned} P_0 &: \ \overline{\pi} \leftarrow \mathsf{PG}(\pi), (c, d) \leftarrow \mathsf{Com}(\pi, \overline{\pi}, m) \\ P_1 &: \ (\overline{\pi}, tk) \leftarrow \mathsf{TPG}(\pi), (c, td) \leftarrow \mathsf{TCom}(\pi), \\ &\quad\ d \leftarrow \mathsf{TDecom}(\pi, \overline{\pi}, c, tk, td, m) \end{aligned}$$

2. Perfect binding*: If* $\pi \leftarrow \mathsf{GPG}(1^\kappa)$ *and* $\overline{\pi} \leftarrow \mathsf{PG}(\pi)$, *then for any* c, m, m', d, d' *it holds except for negligible probability over the coins of* GPG *and* PG, *that if* $\mathsf{Decom}(\pi, \overline{\pi}, c, m, d) = \mathsf{Decom}(\pi, \overline{\pi}, c, m', d') = 1$ *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* $\pi \leftarrow \mathsf{GPG}(1^\kappa)$, $\overline{\pi} \leftarrow \mathsf{PG}(\pi)$, $c \leftarrow \mathsf{Com}(\pi, \overline{\pi}, m)$, $c' \leftarrow_R \mathcal{U}(S[\pi])$.

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 Sect. 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* $\mathsf{H} : \{0,1\}^* \rightarrow \mathcal{C}[\pi]$.

The above property allows an application to set instance parameters as $\overline{\pi} := \mathsf{H}(\mathsf{lbl})$, where string lbl 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 $\mathsf{H}(\mathsf{lbl})$ by sampling $\mathsf{PG}(\pi)$, which makes all these instances perfectly binding, while for all labels corresponding to honest parties the simulator can set $\mathsf{H}(\mathsf{lbl})$ by sampling $\mathsf{TPG}(\pi)$, which makes all these instances equivocable.

In the CKEM application, Sect. 5, the label lbl 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.[12] 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 $\overline{\pi} = \mathsf{H}(x)$ generated in the trapdoored way.

Instantiation. The trapdoor commitment scheme satisfying all properties of Definitions 4.1, 4.2 and 4.3, can be implemented with a "Double Pedersen" commitment in a DDH group $\mathbb{G}$ of order q with covert encoding and RO hash onto the group: Global parameters are $\pi = (g_1, g_2) \leftarrow_{\mathrm{R}} \mathbb{G}^2$, instance parameters are $\overline{\pi} = (h_1, h_2) \leftarrow_{\mathrm{R}} \mathbb{G}^2$, and the commitment is $c = (g_1^d \cdot h_1^m, g_2^d \cdot h_2^m)$ where $d \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ is a decommitment. Trapdoor generators TPG and TCom set resp. $(h_1, h_2) = (g_1^{tk}, g_2^{tk})$ for $tk \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ and $c = (g_1^{td}, g_2^{td})$ for $td \leftarrow_{\mathrm{R}} \mathbb{Z}_q$, and trapdoor decommitment to m opens d s.t. $td = d + tk \cdot m \bmod q$. The security proofs for this construction are deferred to the full version of the paper [22].

4.2 Covert SPHF with PCA-Security

A smooth projective hash function (SPHF) for an NP language $\mathcal{L}$, introduced by Cramer and Shoup [18], 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* $(\mathsf{HKG}, \mathsf{Hash}, \mathsf{PHash})$ *and set family $\mathcal{H}$ indexed by π, where* $\mathsf{HKG}(\pi)$ *outputs* (hk, hp)*, and* $\mathsf{PHash}(x, w, hp)$ *and* $\mathsf{Hash}(x, hk)$ *both compute a hash value v s.t. $v \in \mathcal{H}[\pi]$. Furthermore, this tuple must satisfy the following properties:*

[12] 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 Sect. 4.2.

- 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* $v' \leftarrow_R \mathcal{U}(\mathcal{H}[\pi])$*. 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 [28,43] 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 $k' = \mathsf{Dec}(sk, c')$ 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 $\mathsf{HKG}(\pi)$ outputs (hk, hp), an efficient attacker cannot output $v = \mathsf{Hash}(x, hk)$ even given access to a PCA oracle, which holds the witness w and for any query (hp', v') outputs 1 if $v' = \mathsf{PHash}(x, w, hp')$ 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 Sect. 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:

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

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 Eq. 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:

Definition 4.5. *SPHF for language $\mathcal{L}$ with parameter generation algorithm $\mathsf{PG}_{\mathsf{sphf}}$ and prefix language $\mathcal{L}_{\mathsf{pre}}$ is* One-Way under Plaintext Checking Attack *(OW-PCA) if for any efficient $\mathcal{A}$ 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. $x_L \in \mathcal{L}_{\mathsf{pre}}[\pi]$, $(x_R, w) \leftarrow_R \mathcal{R}_{\mathcal{L}}[\pi, x_L]$, $x \leftarrow (x_L, x_R)$, $(hk, hp) \leftarrow \mathsf{HKG}(\pi)$, 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.

Instantiation. Language $\mathcal{L}^{\mathsf{Com}}[\pi]$ in Eq. 1 has a well-known SPHF which satisfies all properties in Definitions 4.4 and 4.5 for the "Double Pedersen" commitment described in Sect. 4.1: The hash key is $hk = (hk_1, hk_2) \leftarrow_R \mathbb{Z}_q^2$, the projection key is $hp = (g_1)^{hk_1}(g_2)^{hk_2}$, Hash on $x = (\overline{\pi}, m, c)$ for $c = (c_1, c_2)$ sets $v \leftarrow (c_1/h_1^m)^{hk_1}(c_2/h_2^m)^{hk_2}$, and PHash on witness d for x sets $v \leftarrow hp^d$. The security proofs for this SPHF are deferred to the full version of the paper [22].

4.3 Covert Identity Escrow

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

IE Syntax. An Identity Escrow (IE) scheme [34] is an entity authentication scheme with operational assumptions and privacy properties similar to a group signature scheme [16]. Namely, a designated party called a *group manager* (GM) uses a key generation algorithm KG to first generate a group public key *gpk* 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 [10] 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 $\mathsf{CertBlind}$, Ver, Link, as follows:

1. The prover uses a certificate blinding algorithm $\mathsf{CertBlind}$ 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 zero-knowledge proof of knowledge for relation

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

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 $\mathsf{Link}(gpk, bc, rt) = 0$ for all $rt \in \mathsf{RTset}$.

The IE syntax and correctness requirements are formally captured as follows:[13]

Definition 4.6. *An* identity escrow *(IE) scheme is a tuple of efficient algorithms* $(\mathsf{KG}, \mathsf{CG}, \mathsf{CertBlind}, \mathsf{Ver}, \mathsf{Link})$ *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*,* $(cert, v, rt) \leftarrow \mathsf{CG}(msk)$
- *Blinding alg.* $\mathsf{CertBlind}$ *outputs a blinded certificate,* $bc \leftarrow \mathsf{CertBlind}(cert)$
- *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 [34], 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. $\mathsf{Ver}(gpk, bc, v) = 1$, 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* $\mathsf{Ver}(gpk, bc^*, v^*) = 1$ *and* $\mathsf{Link}(gpk, bc^*, rt_i) = 0$ *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.)

[13] More generally, $\mathsf{CertBlind}$ 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 [42].

The IE covertness property strengthens the standard IE property of authentication anonymity [34]. 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 Sect. 2.1). 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.

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 Sect. 5.2.

Definition 4.9. *We call an IE scheme* Σ-protocol friendly *if relation $\mathcal{R}^{\mathsf{IE}}$, Eq. (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 Sect. 6 to realize the group cAKE functionality $\mathcal{F}_{\text{g-cAKE}}$ 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 $\mathsf{Link}(gpk_0, bc, rt_0) = \mathsf{Link}(gpk_1, bc, rt_1) = 1$ *is at most negligible for any efficient $\mathcal{A}$, where* $(msk_b, gpk_b) \leftarrow \mathsf{KG}(1^\kappa)$, $(v_b, cert_b, rt_b) \leftarrow \mathsf{CG}(msk_b)$ *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* $\mathsf{Link}(gpk, bc, rt_0) = \mathsf{Link}(gpk, bc, rt_1) = 1$.

Instantiation. An IE scheme which satisfies Definitions 4.7, 4.8, and 4.9, can be implemented using the Pointcheval-Sanders group signature [42]. (We will refer to this IE instantiation as *PS-IE*.) Sketching it briefly, if $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a

bilinear pairing of type-3 with g ($\hat{g}$) a generator of $\mathbb{G}_1$ ($\mathbb{G}_2$), then (1) KG picks $x, y \leftarrow_R \mathbb{Z}_p$ and sets $msk = (x, y)$ and $gpk = (\hat{X}, \hat{Y}) = (\hat{g}^x, \hat{g}^y)$, (2) $\mathsf{CG}(msk)$ picks $\tilde{\sigma} \leftarrow_R \mathbb{G}_1$, $v \leftarrow_R \mathbb{Z}_p$, sets $\tilde{\omega} = \tilde{\sigma}^{x+y \cdot v}$, and outputs certificate $cert = (\tilde{\sigma}, \tilde{\omega})$, validity witness v, and revocation token $rt = \hat{Y}^v$, (3) $\mathsf{CertBlind}(cert)$ picks $t \leftarrow_R \mathbb{Z}_p$ and outputs $bc = (\tilde{\sigma}^t, \tilde{\omega}^t)$, (4) $\mathsf{Ver}(gpk, bc = (\sigma, \omega), v) = 1$ iff $e(\sigma, \hat{X} \cdot \hat{Y}^v) = e(\omega, \hat{g})$, and (5) $\mathsf{Link}(gpk, bc = (\sigma, \omega), rt) = 1$ iff $e(\sigma, \hat{X} \cdot rt) = e(\omega, \hat{g})$. The full details and security proofs are deferred to the full version of the paper [22].

5 Covert Strong Simulation-Sound Conditional KEM

Conditional Key Encapsulation Mechanism (CKEM) [32] is a KEM counterpart of Witness Encryption (WE) [26] and Conditional Oblivious Transfer (COT) [19]. 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 [7,32,33] 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* [32] if w is extractable from any algorithm distinguishing K_S from random.

Benhamouda et al. [7] 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 [33] extended simulation-sound CKEM of [7] 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 [32] and simulation-soundness of [7] 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 Sect. 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. [33]. 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 $\mathcal{R}^{\mathsf{PS-IE}}$ 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 $x' \in \mathcal{R}^{\mathsf{PS-IE}}$ if $x \in \mathcal{R}^{\mathsf{PS-IE}}$. 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 Sect. 5.1 below we define the covert zero-knowledge strong simulation-sound CKEM, and then in Sect. 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* $(\mathsf{GPG}, \mathsf{Snd}, \mathsf{Rec})$ *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 $\mathsf{P}_{\&\mathsf{Out}}(x)$ 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 Sect. 2.1.

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 $(\mathsf{psTGPG}, \mathsf{psTRec})$ replacing $(\mathsf{TGPG}, \mathsf{TRec})$ 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. $(x, w) \in \mathcal{R}$, $(\mathtt{tr}, K_S, K_R) \leftarrow [\mathsf{Snd}(\pi, x) \leftrightarrow \mathsf{Rec}(\pi, x, w)]$, $K' \leftarrow \{0,1\}^\kappa$.
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:

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

   where
   - $\mathsf{Init}[\mathcal{A}_1](1^\kappa)$ sets $(\pi, td) \leftarrow \mathsf{TGPG}(1^\kappa)$ and $(x, st) \leftarrow \mathcal{A}_1^{\mathsf{TRec}_{\&\mathsf{Out}}(\pi,\cdot,td)}(\pi)$;
   - $\mathsf{Exp}_b[\mathcal{A}_2](\pi, td, x, st)$ outputs $b' = \mathcal{A}_2^{\mathsf{SndMod}_{\&\mathsf{Out}}(b,\pi,x),\mathsf{TRec}_{\&\mathsf{Out}}(\pi,\cdot,td)}(st)$ s.t.
     - $\mathsf{SndMod}_{\&\mathsf{Out}}(1, \pi, x)$ runs $\mathsf{Snd}_{\&\mathsf{Out}}(\pi, x)$;
     - $\mathsf{SndMod}_{\&\mathsf{Out}}(0, \pi, x)$ runs $\mathsf{Snd}(\pi, x)$ 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 [33]. Differences from [33] 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.

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 $(\mathsf{GPG}, \mathsf{PG}, \mathsf{Com}, \mathsf{Decom})$ on msg. space $\mathcal{M}$, with RO-compatible instance parameters with param. space $\mathcal{C}$;
- covert SPHF $(\mathsf{HKG}, \mathsf{Hash}, \mathsf{PHash})$ for $\mathcal{L}^{\mathsf{Com}}[\pi]$;
- CRH H and RO's $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_{\mathsf{Com}}$ 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:

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

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:

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

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

$$
\begin{aligned}
ch &\leftarrow \mathsf{H}_1(x, c) \\
\overline{\pi} &\leftarrow \mathsf{H}_{\mathsf{Com}}(x) \\
a &\leftarrow \mathsf{VRec}(x, ch, z) \\
v_S &\leftarrow \mathsf{Hash}((\pi, \overline{\pi}, c, \mathsf{H}(a)), hk) \\
K_S &\leftarrow \mathsf{H}_2(x, c, z, hp, v_S)
\end{aligned}
$$

Fig. 3. Covert CKEM (in ROM) for any relation $\mathcal{R}$ with a Σ-protocol

5.2 Compiler from Σ-Protocol to Covert CKEM in ROM

Our covert CKEM protocol, shown in Fig. 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 Sect. 4.1, and a *covert and OW-PCA secure SPHF* for language $\mathcal{L}^{\mathsf{Com}}[\pi]$ associated with this commitment scheme, see Sect. 4.2 and Eq. (1). In addition, the compiler uses the ROM, and in particular it assumes that the commitment scheme has *RO-compatible instance parameters*, see Sect. 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*.

Comparison with [32]. Our CKEM construction is a modification of the Σ-to-CKEM compiler of Jarecki [32], 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 Definition 4.5 in Sect. 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 Definition 5.2, which asks that the Sender CKEM challenge is secure in the presence of Receiver CKEM oracle that can be

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:

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

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 $a \leftarrow e(\sigma, \hat{Y}^{r})$, $\overline{\pi} = (h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}(x)$, $(c_1, c_2) \leftarrow (g_1^{d} \cdot h_1^{\mathsf{H}(a)}, g_2^{d} \cdot h_2^{\mathsf{H}(a)})$, $z \leftarrow r + \mathsf{H}_1(x, c_1, c_2) \cdot v \bmod p$, 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 $K_S \leftarrow \mathsf{H}_2((gpk, bc), c_1, c_2, z, hp, v_S)$

Fig. 4. Covert CKEM for Pointcheval-Sanders IE relation $\mathcal{R}^{\mathsf{PS-IE}}$.

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 deferred to the full version of the paper [22]:

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 $\mathcal{L}^{\mathsf{Com}}$ is covert, smooth, and OW-PCA secure.*

Efficient Instantiation. In Fig. 4 we show an instantiation of the generic CKEM from Fig. 3, for relation $\mathcal{R}^{\mathsf{PS-IE}}$ defined by the Covert IE based on Pointcheval-Sanders signatures (i.e. PS-IE), see Sect. 4.3, the "Double Pedersen" trapdoor commitment, see Sect. 4.1, and the associated SPHF, see Sect. 4.2.

6 Construction of Group Covert AKE Protocol

In Figs. 5 and 6 we show algorithms $(\mathsf{KG}, \mathsf{CG}, \mathsf{Auth})$ which implement a generic group cAKE construction from covert Identity Escrow (IE) and covert CKEM. In Fig. 5 we show the group initialization algorithm KG and certificate generation algorithm CG, which implement respectively the GInit and CertInit interfaces of UC group cAKE, as defined in Sect. 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 Fig. 6. Note that if a party is called with command $(\mathsf{NewSession}, \mathsf{ssid}, \perp)$ then it executes as a random beacon, as noted in Fig. 5, instead of following the Auth protocol of Fig. 6.

The authentication protocol Auth in Fig. 6 uses the same combination of IE and CKEM as in the covert AKE of [32], 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 [32], 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 group cAKE notion defined in Sect. 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 Fig. 6 has a minimally-interactive instantiation shown in Fig. 7.

The security of the above group cAKE construction is captured in the following theorem, with a proof deferred to the full version of the paper [22]:

Protocol Ingredients:

- covert IE scheme $\mathsf{IE} = (\mathsf{KG}, \mathsf{CG}, \mathsf{CertBlind}, \mathsf{Ver}, \mathsf{Link})$
- covert CKEM scheme $\mathsf{CKEM} = (\mathsf{GPG}, \mathsf{Snd}, \mathsf{Rec})$ for relation $\mathcal{R}^{\mathsf{IE}}$
- secure channel between GM and each party P

Common Reference String Generation:
Sample CKEM global common reference string $\pi \leftarrow \mathsf{GPG}(1^\kappa)$

Initialization by GM on $(\mathsf{GInit}, \mathsf{gid})$:
Sample $(msk, gpk) \leftarrow_{\mathrm{R}} \mathsf{KG}(1^\kappa)$, broadcast $(\mathsf{gid}, \mathsf{GM}, gpk, \pi)$ to all P's

Certificate Generation by party P on $(\mathsf{CertInit}, \mathsf{gid}, \mathsf{cid})$:
P retrieves $(\mathsf{gid}, \mathsf{GM}, gpk, \pi)$, sends cid to GM on a secure channel
GM sets $(v, cert, rt) \leftarrow \mathsf{CG}(msk)$, sends $(v, cert, rt)$ to P on a secure channel
P stores $(\mathsf{gid}, \mathsf{cid}, v, cert, rt)$, GM stores $T_{rt}[\mathsf{gid}, \mathsf{cid}] \leftarrow rt$

Authenticated Key Exchange by party P on $(\mathsf{NewSession}, \mathsf{ssid}, \mathsf{gid}, \mathsf{cid}, \mathsf{RTcids})$:

1. P retrieves $(\mathsf{gid}, \mathsf{GM}, gpk, \pi)$ and $(\mathsf{gid}, \mathsf{cid}, v, cert, \cdot)$
   P sends $(\mathsf{gid}, \mathsf{RTcids})$ to GM on a secure channel
   GM sends back $\mathsf{RTset} = \{T_{rt}[\mathsf{gid}, \mathsf{cid}]$ s.t. $\mathsf{cid} \in \mathsf{RTcids}\}$ on a secure channel
2. P runs protocol $\mathsf{Auth}((gpk, \pi), (v, cert), \mathsf{RTset})$ in Figure 6
   P outputs $(\mathsf{NewKey}, \mathsf{ssid}, K, \mathsf{cid}_{\mathsf{CP}})$ for $(K, \mathsf{cid}_{\mathsf{CP}})$ output by protocol Auth

Random Beacon implemented by party P on $(\mathsf{NewSession}, \mathsf{ssid}, \perp)$:
Party P runs $\mathsf{Auth}^{\$(\kappa)}$, a random beacon of the same bandwidth as protocol Auth

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

P on $((gpk, \pi), (v, cert), \mathsf{RTset})$		P' on $((gpk, \pi), (v', cert'), \mathsf{RTset}')$
$bc \leftarrow \mathsf{CertBlind}(cert)$	$\xrightarrow{bc}$ $\quad$ $\xleftarrow{bc'}$	$bc' \leftarrow \mathsf{CertBlind}(cert')$
$K_S \leftarrow \mathsf{Snd}(\pi, (gpk, bc'))$	$\longleftrightarrow$ $\boxed{\underset{(\mathsf{P} \to \mathsf{P}')}{\mathrm{CKEM}}}$ $\longleftrightarrow$	$\mathsf{Rec}(\pi, (gpk, bc'), v') \to K'_R$
$K_R \leftarrow \mathsf{Rec}(\pi, (gpk, bc), v)$	$\longleftrightarrow$ $\boxed{\underset{(\mathsf{P} \leftarrow \mathsf{P}')}{\mathrm{CKEM}}}$ $\longleftrightarrow$	$\mathsf{Snd}(\pi, (gpk, bc)) \to K'_S$
If $\exists\ rt \in \mathsf{RTset}$ s.t. $\mathsf{Link}(gpk, bc', rt) = 1$ then $K \leftarrow \perp$, $\mathsf{cid}_{\mathsf{CP}} \leftarrow \mathsf{cid}[rt]$ else $K \leftarrow \mathsf{H}(\{K_S, K_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \perp$		If $\exists\ rt' \in \mathsf{RTset}'$ s.t. $\mathsf{Link}(gpk, bc, rt') = 1$ then $K' \leftarrow \perp$, $\mathsf{cid}'_{\mathsf{CP}} \leftarrow \mathsf{cid}[rt']$ else $K' \leftarrow \mathsf{H}(\{K'_S, K'_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \perp$

H is RO with range $\{0,1\}^\kappa$, notation $\{a, b\}_{\mathrm{ord}}$ stands for $(\min(a, b), \max(a, b))$

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

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 $\mathsf{H}_{\mathsf{Com}}, \mathsf{H}_1, \mathsf{H}_2$ 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 $\pi = (g_1, g_2) \leftarrow_{\mathrm{R}} (\mathbb{G})^2$

Group Manager GM Initialization:

Set $msk = (x, y) \leftarrow_{\mathrm{R}} (\mathbb{Z}_p)^2$ and $gpk = (\hat{X}, \hat{Y}) \leftarrow (\hat{g}^x, \hat{g}^y)$

Certificate Generation by GM:

For P: Pick $(v, u) \leftarrow_{\mathrm{R}} (\mathbb{Z}_p)^2$, set $(\tilde{\sigma}, \tilde{\omega}) \leftarrow (g^u, g^{u(x+y\cdot v)})$, $rt \leftarrow \hat{Y}^v$

For P': Pick $(v', u') \leftarrow_{\mathrm{R}} (\mathbb{Z}_p)^2$, 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 $t \leftarrow_{\mathrm{R}} \mathbb{Z}_p$	$(\sigma', \omega') \leftarrow ((\tilde{\sigma}')^{t'}, (\tilde{\omega}')^{t'})$ for $t' \leftarrow_{\mathrm{R}} \mathbb{Z}_p$
$(r, d) \leftarrow_{\mathrm{R}} \mathbb{Z}_p \times \mathbb{Z}_q$, $a \leftarrow e(\sigma, \hat{Y}^r)$	$(r', d') \leftarrow_{\mathrm{R}} \mathbb{Z}_p \times \mathbb{Z}_q$, $a' \leftarrow e(\sigma', \hat{Y}^{r'})$
$(h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma, \omega)))$	$(h'_1, h'_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma', \omega')))$
$(c_1, c_2) \leftarrow (g_1^d h_1^{\mathsf{H}(a)}, g_2^d h_2^{\mathsf{H}(a)})$	$(c'_1, c'_2) \leftarrow (g_1^{d'} h_1'^{\mathsf{H}(a')}, g_2^{d'} h_2'^{\mathsf{H}(a')})$
$z \leftarrow r + ch \cdot v \bmod p$ for $ch = \mathsf{H}_1(gpk, \sigma, \omega, c_1, c_2)$	$z' \leftarrow r' + ch' \cdot v' \bmod p$ for $ch' = \mathsf{H}_1(gpk, \sigma', \omega', c'_1, c'_2)$
$(hk_1, hk_2) \leftarrow_{\mathrm{R}} (\mathbb{Z}_q)^2$	$(hk'_1, hk'_2) \leftarrow_{\mathrm{R}} (\mathbb{Z}_q)^2$
$hp \leftarrow g_1^{hk_1} g_2^{hk_2}$	$hp' \leftarrow g_1^{hk'_1} g_2^{hk'_2}$
$\xrightarrow{bc = (\sigma, \omega),\ (c_1, c_2, z),\ hp}$	$\xleftarrow{bc' = (\sigma', \omega'),\ (c'_1, c'_2, z'),\ hp'}$
$K_R \leftarrow \mathsf{H}_2(gpk, \sigma, \omega, c_1, c_2, z, (hp')^d)$	$K'_R \leftarrow \mathsf{H}_2(gpk, \sigma', \omega', c'_1, c'_2, z', (hp)^{d'})$
$a' \leftarrow e(\sigma', \hat{X}^{ch'} \hat{Y}^{z'}) \cdot e(\omega', \hat{g}^{-ch'})$ for $ch' = \mathsf{H}_1(gpk, \sigma', \omega', c'_1, c'_2)$	$a \leftarrow e(\sigma, \hat{X}^{ch} \hat{Y}^{z}) \cdot e(\omega, \hat{g}^{-ch})$ for $ch = \mathsf{H}_1(gpk, \sigma, \omega, c_1, c_2)$
$(h'_1, h'_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma', \omega')))$	$(h_1, h_2) \leftarrow \mathsf{H}_{\mathsf{Com}}((gpk, (\sigma, \omega)))$
$v_S \leftarrow (c'_1 h_1'^{-\mathsf{H}(a')})^{hk_1} \cdot (c'_2 h_2'^{-\mathsf{H}(a')})^{hk_2}$	$v'_S \leftarrow (c_1 h_1^{-\mathsf{H}(a)})^{hk'_1} \cdot (c_2 h_2^{-\mathsf{H}(a)})^{hk'_2}$
$K_S \leftarrow \mathsf{H}_2(gpk, \sigma', \omega', c'_1, c'_2, z', v_S)$	$K'_S \leftarrow \mathsf{H}_2(gpk, \sigma, \omega, c_1, c_2, z, v'_S)$
If $\exists\ rt \in \mathsf{RTset}$ s.t. $e(\sigma', \hat{X} \cdot rt) = e(\omega', \hat{g})$	If $\exists\ rt' \in \mathsf{RTset}'$ s.t. $e(\sigma, \hat{X} \cdot rt) = e(\omega, \hat{g})$
then $K \leftarrow \perp$, $\mathsf{cid}_{\mathsf{CP}} \leftarrow \mathsf{cid}[rt]$	then $K' \leftarrow \perp$, $\mathsf{cid}'_{\mathsf{CP}} \leftarrow \mathsf{cid}[rt']$
else $K \leftarrow \mathsf{H}(\{K_S, K_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \perp$	else $K' \leftarrow \mathsf{H}(\{K'_S, K'_R\}_{\mathrm{ord}}), \mathsf{cid}_{\mathsf{CP}} \leftarrow \perp$
Output $(K, \mathsf{cid}_{\mathsf{CP}})$	Output $(K', \mathsf{cid}'_{\mathsf{CP}})$

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

Theorem 6.1. *Protocol* $\Pi = (\mathsf{KG}, \mathsf{CG}, \mathsf{Auth})$ *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 Figs. 5, 6. This instantiation uses the PS-IE scheme based on Pointcheval-Sanders signatures, see Sect. 4.3, and the CKEM from Sect. 5 instantiated as shown in Fig. 4. (See the full version [22] for a walk through this instantiation and an explanation of its steps.) Note that the protocol has 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.

References

1. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_14
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.-C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy (S&P), pp. 180–196 (2003)
3. Bellare, M., Canetti, R., Krawczyk, H.: 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), pp. 419–428 (1998)
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
5. Bellovin, S.M., Merritt, M.: Encrypted key-exchange: password-based protocols secure against dictionary attacks. In: IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72–84 (1992)
6. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_24
7. Benhamouda, F., Couteau, G., Pointcheval, D., Wee, H.: Implicit zero-knowledge arguments and applications to the malicious setting. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 107–129. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_6
8. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: CCS, pp. 967–980. ACM (2013)
9. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
10. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 168–177. ACM Press, October 2004

11. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_7
12. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_5
13. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society (2001)
14. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_22
15. Chandran, N., Goyal, V., Ostrovsky, R., Sahai, A.: Covert multi-party computation. In: FOCS, pp. 238–248. IEEE Computer Society (2007)
16. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_22
17. Cho, C., Dachman-Soled, D., Jarecki, S.: Efficient concurrent covert computation of string equality and set intersection. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 164–179. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_10
18. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
19. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_6
20. Damgård, I.: On $\sum$-protocols (2010). https://cs.au.dk/SIMivan/Sigma.pdf
21. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Crypt. **2**, 107–125 (1992)
22. Eldefrawy, K., Genise, N., Jarecki, S.: Short concurrent covert authenticated key exchange (short cAKE). Cryptology ePrint Archive, Paper 2023/xxx (2023). https://eprint.iacr.org/2023/xxx
23. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_40
24. Fischlin, M.: Trapdoor commitment schemes and their applications. Ph.D. thesis, Goethe University Frankfurt, Frankfurt am Main, Germany (2001)
25. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discret. Appl. Math. **156**(16), 3113–3121 (2008)
26. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Symposium on Theory of Computing Conference, STOC 2013, pp. 467–476. ACM (2013)
27. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206. ACM (2008)
28. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: STOC, pp. 365–377. ACM (1982)

29. Goyal, V., Jain, A.: On the round complexity of covert computation. In: STOC, pp. 191–200. ACM (2010)
30. Gu, Y., Jarecki, S., Krawczyk, H.: KHAPE: asymmetric PAKE from key-hiding key exchange. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 701–730. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84259-8_24
31. Hopper, N.J., Langford, J., von Ahn, L.: Provably secure steganography. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 77–92. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_6
32. Jarecki, S.: Practical covert authentication. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 611–629. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_35
33. Jarecki, S.: Efficient covert two-party computation. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10769, pp. 644–674. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_22
34. Kilian, J., Petrank, E.: Identity escrow. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 169–185. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055727
35. Krawczyk, H.: SKEME: a versatile secure key exchange mechanism for internet. In: 1996 Internet Society Symposium on Network and Distributed System Security (NDSS), pp. 114–127 (1996)
36. Krawczyk, H.: SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_24
37. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
38. Kumar, R., Nguyen, K.: Covert authentication from lattices. In: Ateniese, G., Venturi, D. (eds.) Applied Cryptography and Network Security. ACNS 2022. LNCS, vol. 13269, pp. 480–500. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-09234-3_24
39. Manulis, M., Pinkas, B., Poettering, B.: Privacy-preserving group discovery with linear complexity. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 420–437. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13708-2_25
40. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol (2016). https://signal.org/docs/specifications/x3dh/
41. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_19
42. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_7
43. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–358. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25937-4_22
44. Appelbaum, J., Dingledine, R.: How governments have tried to block Tor. https://oldsite.andreafortuna.org/security/files/TOR/slides-28c3.pdf
45. Sachdeva, A.: DARPA making an anonymous and hack-proof mobile communication system. FOSSBYTES Online Article (2019). https://fossbytes.com/darpa-anonymous-hack-proof-mobile-communication-system/

46. Shbair, W.M., Cholez, T., Goichot, A., Chrisment, I.: Efficiently bypassing SNI-based https filtering. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 990–995 (2015)
47. Tibouchi, M.: Elligator squared: uniform points on elliptic curves of prime order as uniform random strings. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 139–156. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_10
48. Vipin, N.S., Abdul Nizar, M.: Efficient on-line spam filtering for encrypted messages. In: 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), pp. 1–5 (2015)
49. von Ahn, L., Hopper, N.J., Langford, J.: Covert two-party computation. In: STOC, pp. 513–522. ACM (2005)
50. Wahby, R.S., Boneh, D.: Fast and simple constant-time hashing to the BLS12-381 elliptic curve. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(4), 154–179 (2019)