Identity-Concealed Authenticated Encryption and Key Exchange*

Yunlei Zhao

Department of Computer Science, Fudan University, Shanghai, China

Abstract. *Identity concealment* and *zero-round trip time* (0-RTT) *connection* are two of current research focuses in the design and analysis of secure transport protocols, like TLS1.3 and Google's QUIC, in the client-server setting. In this work, we introduce a new primitive for identity-concealed authenticated encryption in the public-key setting, referred to as *higncryption*, which can be viewed as a novel monolithic integration of public-key encryption, digital signature, and identity concealment. We present the security definitional framework for higncryption, and a conceptually simple (yet carefully designed) protocol construction.

As a new primitive, higncryption can have many applications. In this work, we focus on its applications to 0-RTT authentication, showing higncryption is well suitable to and compatible with QUIC and OPTLS, and on its applications to *identity-concealed* authenticated key exchange (CAKE) and unilateral CAKE (UCAKE). In particular, we make a systematic study on applying and incorporating higncryption to TLS. Of independent interest is a new concise security definitional framework for CAKE and UCAKE proposed in this work, which unifies the traditional BR and (post-ID) frameworks, enjoys composability, and ensures very strong security guarantee. Along the way, we make a systematically comparative study with related protocols and mechanisms including Zheng's signcryption, one-pass HMQV, QUIC, TLS1.3 and OPTLS, most of which are widely standardized or in use.

1 Introduction

Identity concealment and *zero-round trip time (0-RTT) option* are two of current research focuses in design and analysis of cryptographic systems (particularly, secure transport protocols for the client-server setting). By identity concealment, we mean that the transcript of protocol run should not leak participants' identity information, which is now deemed to be an important privacy concern and is mandated or recommended by a list of widely standardized and deployed cryptographic protocols like TLS1.3 [16], QUIC [40], EMV [8], etc. Furthermore, informally speaking, a player enjoys *forward ID-privacy* if its ID-privacy preserves even when its static secret-key is compromised. By 0-RTT option, we mean that when the client has a previously retrieved or cached public key of the server, it can optionally transmit encrypted information already in the first flow of the protocol run. 0-RTT connection is highly desirable because of its significant impact on connection latency, a critical issue in most HTTP(S) content acquisitions. This

* Extended abstract of this work appeared at CCS 2016: 1464-1479.

option is supported by QUIC and is now under discussion by the IETF TLS1.3 working group.

The QUIC protocol, developed by Google and already implemented with Chrome in 2013, currently stands as one of the most promising solutions to decreasing latency while intending to provide security properties similar with TLS [30]. According to Google's measurement, currently at least 75% of all QUIC connections use 0-RTT mode. Unfortunately, QUIC and TLS1.3 (which is underway) are now only supporting 0-RTT mode *without* client authentication. One of the major obstacles (among others) hindering the deployment of 0-RTT client authentication, from our point of view, is that the literature lacks a cryptographic mechanism that soundly and practically integrates public-key encryption, entity authentication and ID-concealment into a single primitive. It has become a consensus that, in order for QUIC to be ubiquitous, a suitable mechanism for 0-RTT client authentication is needed.

A straightforward solution for 0-RTT client authentication is to encrypt client's 0-RTT data and signature using server's public-key. However, this approach has several drawbacks and is unsatisfactory.

- Firstly, such a direct composition may not be sound enough and may bring some security concerns. For example, consider the DHIES-based solution: $\{X = g^x, Enc_K(m, \mathsf{sig})\}$, where the receiver has public-key $B = g^b$, Enc is a CCA-secure symmetric encryption with K being derived from $CDH(X, B)$, sig is sender's signature on message m. If an adversary learns (m, sig), or learns the DH-exponent x (*that is allowed to be exposed in our security model*), it can re-encrypt (m, sig), potentially leading to sender impersonation or causing two un-matching sessions to share the same 0-RTT data and peer view. The underlying reason is that the composition of ID-concealment, encryption and entity authentication is quite loose.
- Secondly, directly composing public-key encryption and digital signature may not be efficiency economic. On the other hand, a tailored protocol construction usually not only has efficiency improvements, but also can have much more advantageous features, as witnessed by the ongoing CAESAR competition [9] on authenticated encryption (AE) even though AE can be generally achieved by composing CPA-secure encryption and MAC.
- Thirdly, viewing the cryptographic mechanism, which integrates ID-concealment, public-key encryption and entity authentication, as a separate primitive may conceptually simplify the design and analysis of complex protocols. However, the proper modeling of ID-concealed authenticated encryption and key-exchange may not be so obvious and deserves explorations, as already witnessed by the modeling of composition of encryption and authentication [1,2,14,20,21].

– Fourthly, sender's signature leaves to the receiver an undeniable proof of session participance, while in many application scenarios certain kind of deniability is more desirable for privacy considerations.

In the public-key setting, authenticated encryption refers to signcryption [48], which is also standardized by ISO-29150. It is shown that signcryption is functionally equivalent to one-pass authenticated key-exchange [35,13,20], which in turn has applications in asymmetrical key-wrapping [21]. Zheng's signcrytion [48,2,17] and one-pass HMQV (HOMQV) [24,21] are briefly described in Appendix A. There, it is clear that sender's public identity information (including its certificate) has to be sent in clear, as otherwise the receiver cannot derive the shared key. It would be interesting to note that, even though signcryption (1997) and one-pass MQV (1995) have been invented for about two decades, the issue of ID concealment was not considered for them up to now, whether for protocol construction or for security definition. It is also interesting to note that HOMQV enjoys "receiver deniability", in the sense that the session transcript (in particular, the authentication value σ) can be simulated from public parameters and receiver's secret-key. In comparison, the session transcript of Zheng's signcryption is undeniable, as the authentication value s corresponding to sender's signature cannot be generated by the receiver. In addition, Zheng's signcryption suffers from the x-security defined in [21]:[1] the leakage of the DH-exponent x causes the exposure of sender's static secret-key a or the pre-shared secrecy PS (for Zheng's signcryption, both a and PS are exposed). This leads us to the following motivation question.

Motivating Question 1

Can we come up with a cryptographic mechanism, simultaneously enjoying: (1) forward ID-privacy, (2) being relatively as efficient as HOMQV, (3) receiver deniability, and (4) x-security?

Authenticated key-exchange (AKE), in particular Diffie-Hellman (DH), plays a fundamental role in modern cryptography, bridging public-key cryptography and symmetric-key cryptography, and is the backbone of a list of network security protocols that are widely standardized and used. Up to now, the most efficient AKE protocols are (H)MQV [35,26] and OAKE [46], but none of them considers ID-concealment. For AKE protocols in the client/server setting, both TLS1.3 and QUIC mandate ID-concealment. The protocol structure of QUIC, briefly described in Appendix A, is conceptually simple and enjoys the advantages of efficiency, receiver deniability, and deployment flexibility. However, QUIC (without 0-RTT connection) does not enjoy forward ID-privacy. Specifically, the compromising of server's static secret-key will expose the shared-key K_1 for encrypting

[1] This is actually named as y-security in [21], where the player pid_B plays the role of sender.

server's DH-component Y, consequently losing server's ID-privacy. The core authentication mechanism of TLS1.3 is based on the SIGMA scheme [25], which is also briefly described in Appendix A. For presentation simplicity, by TLS1.3 we mean the core authentication mechanism presented there. TLS1.3 uses signature for server authentication, which brings the following effects. On the one hand, when implementing with DSA, it can be less efficient than QUIC, and suffers from the shortcoming of DSA-type signature (i.e., the exposure of random nonce, which can be offline generated and stored for online signature generation, will cause exposure of static secret-key). On the other hand, the use of signature loses receiver deniability. Since its sixth version, TLS1.3 incorporates the OPTLS protocol [28], which is actually an dual version of basic mechanism of the deniable IKE (DIKE) protocol proposed in [45,47]. Both DIKE and OPTLS are also described in Appendix A. Briefly speaking, DIKE/OPTLS is as efficient as QUIC, but enjoys forward ID-privacy. But none of QUIC, TLS1.3 and DIKE/OPTLS enjoys x-security, in the sense that the exposure of DH-exponent in a session exposes the session-key (for QUIC and TLS1.3) or server's ID (for OPTLS). By comparison, (H)MQV and OAKE have x-security. This leads to the following motivation question.

Motivating Question 2

Can we come up with new AKE schemes, simultaneously enjoying: (1) forward ID-privacy, (2) efficiency comparable to (H)MQV/OAKE and even better than QUIC/OPTLS, (3) x-security, (4) receiver deniability, and (5) being free of signatures?

1.1 Contributions

In this work, we systematically solve the above two motivating questions, with novel ID-concealed authentication mechanism and new security definitional frameworks.

For the first motivating question, we introduce a new primitive, referred to as identity-*hi*ding si*gncryption* (*higncryption*, for short). We present the security definitional framework for higncryption, and a practical and conceptually simple (yet carefully designed) construction of higncryption, with detailed comparisons with Zheng's signcryption and HOMQV. We suggest that, as a new primitive, higncryption is of independent value and can have many applications. A direct application of higncryption is one-pass ID-concealed AKE, which in turn has application to key-wrapping. We make in-depth discussions on its applications to 0-RTT authentication (showing higncryption is well suitable and compatible to QUIC and OPTLS), and to server-only authenticated and confidential channel establishment (SACCE) as defined in [27]. Compared to QUIC and TLS1.3 as described in Appendix A, the higncryption-based SCAAE is more efficient, signatureless, has forward ID-privacy and receiver deniability, strong resilience to

exposure of intermediate state, and enjoys flexible implementations and deployments.

For the second motivating question, we first observe that existing security definitional frameworks for AKE cannot be well applied to ID-concealed authenticated key-exchange, on the following grounds. On the one hand, traditional AKE security definition critically relies upon users' identities to define session matching, which lies at the heart of AKE security definition. On the other hand, existing security frameworks usually do not consider ID-privacy or treat it separately from AKE security. This motivates us to present a new security definition framework, for both ID-concealed key-exchange with mutual authentication (referred to as CAKE for presentation simplicity) and ID-concealed key-exchange with unilateral authentication (referred to as UCAKE), and make in-depth discussions and clarifications. Then, we present protocol constructions for both CAKE and UCAKE, along with detailed discussions and comparisons with QUIC, TLS1.3 and HMQV.

All the protocols developed in this work are provably secure under standard assumptions in the random oracle model. In order to support more efficient and flexible deployments while still preserving provable security, we introduce a new family of problems and assumptions related to a variant of the DL-problem, referred to as flexible DL (FDL) problem, which are proven to hold in the generic group model and might be of independent interest (e.g., to leakage-resilient cryptography).

2 Preliminaries

If $\mathcal{S}$ is a finite set then $|\mathcal{S}|$ is its cardinality, and $x \leftarrow \mathcal{S}$ is the operation of picking an element uniformly at random from $\mathcal{S}$. If $\mathcal{S}$ denotes a probability distribution, $x \leftarrow \mathcal{S}$ is the operation of picking an element according to $\mathcal{S}$. We overload the notion for probabilistic or stateful algorithms, writing $C \leftarrow Alg$ to mean that algorithm Alg runs and outputs value named C. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. A string or value α means a binary one, and $|\alpha|$ is its binary length. For two strings $x, y \in \{0,1\}^*$, $x||y$ denotes their concatenation.

Let G' be an abelian group of order N, and $G = \langle g \rangle$ be a unique subgroup of G' generated by the generator g of prime order q. Throughout this work, the group law is written multiplicatively, and the length of q, i.e., $|q|$, serves as the security parameter. Denote by 1_G the identity element of G', by $G \setminus 1_G$ the set of elements of G except 1_G, and by $t = N/q$ the cofactor value. When instantiated with groups based on elliptic curves, G' is the group of points $E(L)$ on an elliptic curve E defined over a finite field L, and G is a subgroup of $E(L)$ of prime order q. For elliptic curve based groups, the cofactor t is typically very small.

The discrete logarithm (DL) assumption over G says that given $X = g^x$, for $x \leftarrow Z_q^*$, no probabilistic polynomial-time (PPT) DL-solver algorithm can output x with non-negligible probability. The computational Diffie-Hellman (CDH) assumption says that given $X = g^x$, $Y = g^y$, for $x, y \leftarrow Z_q^*$, no probabilistic polynomial-time CDH-solver algorithm can compute $CDH(X, Y) = g^{xy}$ with non-negligible probability. The Gap Diffie-Hellman (GDH) assumption [36] says that the CDH assumption holds, even if the CDH solver is equipped with a decisional Diffie-Hellman (DDH) oracle for G and g, where on arbitrary input $(U, V, Z) \in G^3$ the DDH oracle outputs 1 if and only if $Z = CDH(U, V)$.

2.1 Authenticated Encryption with Associated Data

Briefly speaking, an *authenticated encryption with associated data* (AEAD) scheme transforms a message M and a public header information H (e.g., a packet header, an IP address, etc) into a ciphertext C in such a way that C provides both privacy (of M) and authenticity (of C and H) [38]. In practice, when AEAD is used within cryptographic systems, the associated data is usually implicitly determined from the context (e.g., the hash of the transcript of protocol run or some pre-determined states).

AEAD SECURITY. Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme. The probabilistic polynomial-time algorithm K_{se} takes a security parameter κ as input and samples a key K from a finite and non-empty set $\mathcal{K} \bigcap \{0,1\}^\kappa$. For presentation simplicity, we assume $K \leftarrow \mathcal{K} = \{0,1\}^\kappa$. The polynomial-time encryption algorithm $\mathsf{Enc} : \kappa \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ and the (deterministic) polynomial-time decryption algorithm $\mathsf{Enc} : \kappa \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ satisfy: for any $K \leftarrow \mathcal{K}$, any associate data $H \in \{0,1\}^*$ and any message $M \in \{0,1\}^*$, if $\mathsf{Enc}_K(H, M)$ outputs $C \neq \perp$, then $\mathsf{Dnc}_K(C)$ always outputs M. Here, we assume the ciphertext C bears the associate data H in plain.

Let A be an adversary. Table 1 describes a security game for AEAD. We define the advantage of A to be $\mathbf{Adv}^{\text{aead}}_{\mathsf{SE}}(A) = \left|2 \cdot \Pr[\text{AEAD}^A_{\mathsf{SE}} \ \textit{returns} \ \mathsf{true}] - 1\right|$. We say that the SE scheme is AEAD-secure, if for all sufficiently large κ the advantage of any probabilistic polynomial-time adversary is negligible.

main $\text{AEAD}^A_{\mathsf{SE}}$:	**procedure** $\mathbf{Enc}(H, M_0, M_1)$:	**procedure** $\mathbf{Dec}(C')$:
$K \leftarrow \mathcal{K}_{\text{se}}$	If $\lvert M_0\rvert \neq \lvert M_1\rvert$, Ret $\perp$	If $\sigma = 1 \wedge C' \notin \mathcal{C}$ then
$\sigma \leftarrow \{0,1\}$	$C_0 \leftarrow \mathsf{Enc}_K(H, M_0)$	Ret $\mathsf{Dec}_K(C')$
$\sigma' = A^{\mathbf{Enc},\mathbf{Dec}}$	$C_1 \leftarrow \mathsf{Enc}(H, M_1)$	Ret $\perp$
Ret $(\sigma' = \sigma)$	If $C_0 = \perp$ or $C_1 = \perp$, Ret $\perp$	
	$\mathcal{C} \xleftarrow{\cup} C_\sigma$; Ret C_σ	

Table 1: AEAD security game

The above AEAD security is quite strong. In particular, it means that, after adaptively seeing a polynomial number of ciphertexts, an efficient adversary is infeasible to generate a new valid ciphertext in the sense its decryption is not "$\perp$". Also, for two independent keys $K, K' \leftarrow \mathcal{K}_{se}$ and any message M and any header information H, $\Pr[\mathsf{Dec}_{K'}(\mathsf{Enc}_K(H, M)) \neq \perp]$ is negligible.

The AEAD security definition is based on that in [39,37], with the following modifications: the length-hiding requirement is removed while header information integrity property is added. In this work, we assume users' identities and public-key information to be of equal length; otherwise, we need length-hiding AEAD as defined in [37,27].

In general, AEAD schemes can be built upon the composition of a CPA-secure symmetric encryption scheme and a MAC scheme [3]. But until recently, authenticated encryption has been recognized as an important *distinct* cryptographic primitive both in cryptographic research and in cryptographic standardizations (e.g., the ongoing CAESAR competition [9]). The reasons are as follows [14]. Firstly, there are inappropriate composition of symmetric encryption and MACs that result in insecure AEAD schemes [3,24]. Secondly, there are constructions of AEAD that escape the generic composition paradigm, and in most cases a tailored solution can be noticeably more efficient or have other advantages compared to the generic composition paradigm. Finally, recognizing authenticated encryption as a distinct cryptographic primitive may conceptually simplify the design of complex protocols that require both privacy and authenticity, as witnessed by the design and analysis of secure channel establishment with AEAD in TLS/SSL [27], QUIC [30], EMV [8], etc. Currently, the most popular AEAD scheme in use may be GCM-AES [34].

2.2 General Forking Lemma

Lemma 1 (General forking lemma [4]). *Fix an integer $\hat{q} \geq 1$ and a set $\mathcal{O}$ of size λ. Let $\mathcal{C}$ be a randomized algorithm that on input $U, d_1, \cdots, d_{\hat{q}}$ returns a pair, the first element of which is an integer in the range $0, \cdots, \hat{q}$ and the second element of which we refer to as a* side output*. Let IG be a randomized algorithm that we call the input generator. The accepting probability of $\mathcal{C}$, denoted acc, is defined as*

$$\Pr[J \geq 1 : U \leftarrow IG; d_1, \cdots, d_{\hat{q}} \leftarrow \mathcal{O}; (J, \sigma) \leftarrow \mathcal{C}(U, d_1, \cdots, d_{\hat{q}})]$$

The forking algorithm $F_{\mathcal{C}}$ associated with $\mathcal{C}$ is the randomized algorithm that takes input U and proceeds as follows:

Algorithm 1 The Forking Algorithm $F_{\mathcal{C}}(U)$

Pick coins ρ for $\mathcal{C}$ at random
$d_1, \cdots, d_{\hat{q}} \leftarrow \mathcal{O}$
$(J, \sigma) \leftarrow \mathcal{C}(U, d_1, \cdots, d_{\hat{q}}; \rho)$
if $J = 0$ **then**
    return $(0, \perp, \perp)$
end if
$d'_J, \cdots, d'_{\hat{q}} \leftarrow \mathcal{O}$
$(J', \sigma') \leftarrow \mathcal{C}(U, d_1, \cdots, d_{J-1}, d'_J, \cdots, d'_{\hat{q}}; \rho)$
if $J = J' \wedge d_J \neq d'_J$ **then**
    return $(1, \sigma, \sigma')$
else
    return $(0, \perp, \perp)$
end if

Let

$$\mathrm{frk} = \Pr\left[b = 1 : U \leftarrow \mathsf{IG}; (b, \sigma, \sigma') \leftarrow F_{\mathcal{C}}(U)\right].$$

Then

$$\mathrm{frk} \geq \mathrm{acc}\left(\frac{\mathrm{acc}}{\hat{q}} - \frac{1}{\lambda}\right). \qquad \square$$

3 Flexible Discrete Logarithm (FDL) and Related Problems

In this section, we introduce a class of problems that generalize the traditional DL and related problems, study their complexity in the generic group model, and discuss their applications and consequences.

We first prove the following lemma, which could be viewed as a generalized version of the Schwartz-Shoup lemma [42,43,33,31].

Lemma 2. *Let $\mathcal{X}_i$, $1 \leq i \leq k$, be a* well-spread *distribution over a subset $\mathcal{S}_i \subseteq Z_q^*$ with min-entropy $\lambda_i = -log(\max_{x \in \mathcal{S}_i}(\Pr[\mathcal{X}_i = x])) > \omega(\log |q|)$, which means $\Pr[\mathcal{X}_i = x]$ is negligible for any $x \in Z_q$ (when the security parameter $|q|$ is sufficiently large). Let $P(X_1, \cdots, X_k)$ be a* non-zero *multivariate polynomial over Z_q of total degree d, where d and k are polynomials in $|q|$ (usually, they are small constants). The probability $P(x_1, \cdots, x_k) = 0$, when x_i taken independently from $\mathcal{X}_i$ (i.e., $x_i \leftarrow \mathcal{X}_i$) for $1 \leq i \leq k$, is at most $d(2^{-\lambda_1} + \cdots + 2^{-\lambda_k})$.*

Proof. We prove this lemma by induction on k. Firstly, a univariate polynomial (i.e., for the case of $k = 1$) over the Z_q has at most d roots, and thus $\Pr[P(x_1) = 0] \leq d2^{-\lambda_1}$ for $x_1 \leftarrow \mathcal{X}_1$.

Secondly, consider the case of $k = 2$. Let $e \geq 1$ be the maximal degree of x_2 in any term in $P(x_1, x_2)$. The polynomial $P(x_1, x_2)$ can be viewed as a univariate

polynomial of degree e: $P_2(x_2) = c_e x_2^e + c_{e-1} x_2^{e-1} + \cdots + c_1 x_2 + c_0$, where c_i, $1 \leq i \leq e$, is a univariate polynomial in x_1 of degree at most $d-i$, and c_e is non-zero. There are at most $d-e$ solutions of x_1 for $c_e = 0$. Accordingly, there are at most $d-e$ solutions of x_1 for $c_e = c_{e-1} = \cdots = c_0 = 0$, which causes P_2 to be a zero polynomial. This event occurs with probability $(d-e)2^{-\lambda_1}$. On the other hand, if this event does not occur (i.e., for any value x_1 under which $P_2(x_2)$ is non-zero), there are at most e solutions of x_2 for $P_2(x_2) = 0$. Thus, for $x_1 \leftarrow \mathcal{X}_1$ and $x_2 \leftarrow \mathcal{X}_1$, $\Pr[P(x_1, x_2) = 0] \leq (d-e)2^{-\lambda_1} + (1-(d-e)2^{-\lambda_1})e2^{\lambda_2} \leq (d-e)2^{-\lambda_1} + e2^{\lambda_2} < d(2^{-\lambda_1} + 2^{-\lambda_2})$.

Now, supposing this lemma holds for case of $k-1$, we consider the case of k. Let $\vartheta \geq 1$ be the maximal degree of x_k in any term in $P(x_1, \cdots, x_k)$. The polynomial $P(x_1, \cdots, x_k)$ can be viewed as a univariate polynomial $P_k(x_k)$ of degree ϑ, where the coefficient of the term x^ϑ is a polynomial $e_\vartheta(x_1, \cdots, x_{k-1})$ of degree at most $d-\vartheta$. According to the assumption, $\Pr[e_\vartheta(x_1, \cdots, x_{k-1}) = 0] \leq (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}}))$, which is also the probability upper-bound for causing P_k to be a zero polynomial. On the other hand, conditioned on this event does not occur (i.e., for all values $(x_1, \cdots, x_k)$ under which P_k is non-zero), there are at most ϑ solutions of x_k for $P_k(x_k) = 0$. As $\lambda_i > \omega(\log |q|)$, $1 \leq i \leq k$, and d and k are polynomial in $|q|$, we have $d(2^{-\lambda_1} + \cdots + 2^{-\lambda_k} < 1$ (when $|q|$ is sufficiently large). In summary, for $x_i \leftarrow \mathcal{X}_i$, $1 \leq i \leq k$, $\Pr[P(x_1, \cdots, x_k) = 0] \leq (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}})) + (1-(d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}})))\vartheta 2^{\lambda_k} \leq (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}})) + \vartheta 2^{\lambda_k} < d(2^{-\lambda_1} + \cdots + 2^{-\lambda_k})$. □

Let $\mathcal{X}$ and $\mathcal{Y}$ (resp., $\mathcal{Z}$) be well-spread distributions over some subsets of of Z_q^* (resp., the uniform distribution over Z_q^*) with min-entropy $\lambda_\mathcal{X}$ and $\lambda_\mathcal{Y}$ respecitvely. The flexible discrete logarithm (FDL) problem is to compute x from $X = g^x$ for $x \leftarrow \mathcal{X}$. The flexible CDH (FCDH) problem is to compute $CDH(X = g^x, Y^y)$ for $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$. The flexible DDH (FDDH) problem is to distinguish $(X = g^x, Y = g^y, g^{xy})$ and (X, Y, g^z), for $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$ and $z \leftarrow Z_q^*$. The flexible gap Diffie-Hellman (FGDH) problem is to compute $CDH(X = g^x, Y = g^y)$ for $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$ with the aid of a DDH oracle. The flexible gap DL (FGDL) problem is to distinguish between g^{xy} and g^z, for $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$ and $z \leftarrow Z_q^*$, with the aid of a DDH oracle. Notice the difference between FGDL and FDDH, where for solving FGDL the values g^x and g^y are not given as input.

Clearly, traditional problems of DL, CDH, DDH, GDH are special cases of their flexible counterparts, when $\mathcal{X}$ and $\mathcal{Y}$ are constrained to the uniform distribution over Z_q^* with min-entropy $\log(q-1)$. The work [10] introduces a variant of the DDH problem, referred to as hybrid DDH (HDDH) for presentation simplicity, where $\mathcal{X}$ is a well-spread distribution over Z_q^* while $\mathcal{Y}$ is the uniform distribution over Z_q^*, which is a special case of FDDH and can be viewed as a hybrid of FDDH and traditional DDH. Similarly, we can define hybrid CDH (CDH), resp., hybrid GDH

(HGDH), where $\mathcal{X}$ is a well-spread distribution over Z_q^* while one of $\mathcal{X}$ and $\mathcal{Y}$ is the uniform distribution over Z_q^*. Actually, as shall see, what we need in this work, *only for flexible efficient deployment of protocols*, are the HGDH assumption and the FGDL assumption, where FGDL assumption is used only for proving forward ID-privacy of the flexible implementations.

Next, we study the complexity of FDL and related problems in the generic group model [43,31]. Roughly speaking, an algorithm is generic if it does not use the encoding of the group elements. It can only use group elements for group operations and relation verifications. There are many groups for which the fastest DL solver algorithms are generic. For example, general elliptic curves; general hyper-elliptic curves of genus 2; and subgroups of prime order q in Z_p^* when $(p-1)/q$ is so large that sieving methods are inefficient [41].

For presentation simplicity, in the following analysis we use Maurer's generic group model [31] that is actually equivalent to Shoup's model [43,22], and only count the complexity of generic steps where each generic step corresponds to an access to the generic group oracle for performing one group operation or one relationship verification. Also, we disregard the probability of simply and correctly guessing x or y, which happens with probability of $\max\{2^{-\lambda_x}, 2^{-\lambda_y}\}$ that is negligible.

Theorem 1. *For an algorithm of τ generic steps for solving the FGDH problem, its success probability is upper bounded by $\tau^3(2^{-\lambda_x} + 2^{-\lambda_y})$ in the generic group model.*

Proof. In Maurer's generic group model for solving the FGDH problem, the generic group oracle (GG-oracle) $\mathcal{O}$ originally keeps three internal states $(1, x, y)$ in a list L, where x (resp., y) is taken independently according to the well-spread distribution $\mathcal{X}$ (resp., $\mathcal{Y}$). For presentation simplicity, we denote by $L[i]$ the value stored in the i-th entry of L, and we assume $L[1] = 1$, $L[2] = x$ and $L[3] = y$. The adversary is given the indices of $(1, x, y)$ in L, i.e., $(1, 2, 3)$, and has black-box access to the GG-oracle $\mathcal{O}$. For the i-th GG-oracle access corresponding to a group operation, the value computed by the GG-oracle $\mathcal{O}$ can be viewed as a linear polynomial of the form $F_i(x, y) = a_i x + b_i y + c_i$, where $a_i, b_i, c_i \in Z_q$ are determined by previous GG-oracle accesses. The value F_i is not returned to $\mathcal{A}$ directly, but is stored into a position in the internal list L where the position index for storing F_i is indicated by $\mathcal{A}$. $\mathcal{A}$ is always given the ability of verifying equality relation, by which $\mathcal{A}$ queries $\mathcal{O}$ with (i, j) and gets result whether $L[i] = L[j]$ or not. For adversary against FGDH problem, the adversary $\mathcal{A}$ is additionally allowed to query the GG-oracle with (i, j, k), and GG-oracle returns 1 if and only if $L[i]L[j] = L[k]$ which corresponds to the DDH oracle.

As discussed in [31], in this generic group model we only need to consider non-adaptive adversaries, and there are only two approaches for $\mathcal{A}$ to succeed (other

than simply guessing x or y that is disregarded in the analysis). One is to cause two different F_i and F_j to collide, in the sense that $a_ix + b_iy + c_i = a_jx + b_jy + c_j$ where $(a_i, b_i, c_i) \neq (a_j, b_j, c_j)$. In other words, $(a_i - a_j)x + (b_i - b_j)y + (c_i - c_j) = 0$. By Lemma 2, this event can occur with probability at most $C_\tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$.

Another approach for $\mathcal{A}$ to succeed is to cause, for some (i, j, k), the *non-zero* polynomial $F_iF_j - F_k = 0$ (if the polynomial $F_iF_j - F_k$ is a zero-polynomial, it leaks nothing). That is, $(a_ix + b_iy + c_i)(a_jx + b_jy + c_j) - (a_kx + b_ky + c_i) = 0$ for $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$. Note that $F_iF_j - F_k$ is a quadratic polynomial. According to Lemma 2 by setting $d = 2$, this event occurs with probability at most $C_\tau^3(2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}}))$.

Note that $C_\tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}}) + C_\tau^3(2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})) < \tau^3(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$. □

Corollary 1. *For an algorithm of τ generic steps for solving the FCDH, resp., FDL, problem, its success probability is upper bounded by $\tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$, resp., $\tau^2 2^{-\lambda_\mathcal{X}}$, in the generic group model.*

Proof. For solving FCDH problem in the generic group model, as no DDH oracle is rendered to the adversary, $\mathcal{A}$ can only perform equality verification via oracle access (besides group operations). In this case, we only need to consider the probability that two different polynomials $F_i = a_ix + b_iy + c_i$ and $F_j = a_jx + b_jy + c_j$ collide for $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$. This event occurs with probability at most $\tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$.

For solving FDL problem in the generic group model, the GG-oracle $\mathcal{O}$ originally keeps $(1, x)$ in its internal list, where $x \leftarrow \mathcal{X}$. The adversary succeeds only when two different polynomials $F_i = a_ix + c_i$ and $F_j = a_jx + c_j$ collide. This event occurs with probability at most $\tau^2 2^{-\lambda_\mathcal{X}}$ according to Lemma 2. □

Theorem 2. *For an algorithm of τ generic steps for solving the FDDH problem, its success probability is upper bounded by $\frac{1}{2} + \tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})$ in the generic group model.*

Proof. In Maurer's generic group model for solving the FDDH problem, the GG-oracle $\mathcal{O}$ originally keeps $(1, x, y, T_0, T_1)$ in its internal list, where x (resp., y) is taken independently according to the well-spread distribution $\mathcal{X}$ (resp., $\mathcal{Y}$), $z \leftarrow Z_q^*$ and $T_b = xy$ and $T_{1-b} = z$ for a random bit $b \leftarrow \{0, 1\}$. The goal of the adversary $\mathcal{A}$ is to guess the random bit b.

For the i-th GG-oracle access corresponding to a group operation, the value computed by $\mathcal{O}$ can be viewed as a quadratic polynomial of the form $F_i(x, y, xy, z) = a_ix + b_iy + c_ixy + d_iz + e_i$, where $a_i, b_i, c_i, d_i, e_i \in Z_q$ that are determined by previous GG-oracle accesses. Define $G_i(x, y, z) = F_i(x, y, xy, z)$. The advantage obtained by $\mathcal{A}$ (over simply guessing the random bit b) is the probability of making two different polynomials $G_i(x, y, z)$ and $G_j(x, y, z)$ colliding. The non-zero polynomial $G_i(x, y, z) - G_j(x, y, z)$ is of degree 2. According to Lemma 2, $\mathcal{A}$ succeeds with probability at most $\frac{1}{2} + \tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})$. □

Theorem 3. *For an algorithm of τ generic steps for solving the FGDL problem, its success probability is upper bounded by $\frac{1}{2}+\tau^3(2^{-\lambda_x}+2^{-\lambda_y}+\frac{1}{q-1})$ in the generic group model.*

Proof. In Maurer's generic group model for solving the FGDL problem, the GG-oracle $\mathcal{O}$ originally keeps $(1, T_0, T_1)$ in its internal list for $T_b = xy$ and $T_{1-b} = z$, where $b \leftarrow \{0,1\}$, x (resp., y) is taken independently according to the well-spread distribution $\mathcal{X}$ (resp., $\mathcal{Y}$) and $z \leftarrow Z_q^*$. Note that $\mathcal{O}$ does not directly keep the value x or y. The goal of the adversary $\mathcal{A}$ is to guess the random bit b, with the aid of a DDH oracle.

For the i-th GG-oracle access corresponding to a group operation, the value computed by $\mathcal{O}$ can be viewed as a quadratic polynomial of the form $F_i(xy, z) = a_i xy + b_i z + c_i$, where $a_i, b_i, c_i \in Z_q$ that are determined by previous GG-oracle accesses. Define $G_i(x, y, z) = F_i(xy, z)$. The advantage obtained by $\mathcal{A}$ (over simply guessing the random bit b) is the probability of: (1) for some (i, j), making two different quadratic polynomials $G_i(x, y, z)$ and $G_j(x, y, z)$ colliding (i.e., $G_i - G_j = 0$); or (2) for some (i, j, k), making the (non-zero) *quartic* polynomial $G_i G_j - G_k = 0$. According to Lemma 2, the advantage is at most $\frac{1}{2}+C_\tau^2(2(2^{-\lambda_x}+2^{-\lambda_y}+\frac{1}{q-1}))+ C_\tau^3(4(2^{-\lambda_x}+2^{-\lambda_y}+\frac{1}{q-1})) < \frac{1}{2}+\tau^3(2^{-\lambda_x}+2^{-\lambda_y}+\frac{1}{q-1})$. □

Remark. The complexity of FDL and related problems shows that they are hard for polynomial-time algorithms at least in the generic group model. We remark that the upper-bounds proved in the above theorems and corollaries are quite loose, where we have given the solver algorithms the benefit of the doubt, and have assumed that the algorithm will succeed with any collision. In a related work by Schnorr [41], it is shown that if x is drawn uniformly at random from a subset $H \subset Z_q^*$ of size $|H| \leq \sqrt{|q|}$, the success probability upper-bound of a τ generic-step DL-solver is about $\frac{\tau}{|H|}$ (not $\frac{\tau^2}{|H|}$ as established with usual analysis), which implies that DL defined over a random subset of size $\sqrt{|q|}$ is as hard as traditional DL defined over Z_q^*! But the result [41] critically relies on uniform distribution over the subset H, while our result is for any distribution with super-logarithmic min-entropy.

The result about FDL and related problems has two consequences. On the one hand, it indicates that cryptosystems based on the traditional DL and related problems have strong resilience to randomness leakage, in the sense they are still secure as long as super-logarithmic min-entropy remains with each exponent secrecy. On the other hand, it allows more flexible and efficient implementations of cryptosystems based on DL and related problems. For example, user's static secret-key is still drawn uniformly at random from Z_q^*, while the ephemeral DH-exponents could be taken in different (super-polynomial size) subsets of Z^* according to the task criticality and application scenarios.

4 Strong Security Model for Highncryption

An identity-*hi*ding si*gn* en*cryption* (*highncryption*) scheme $\mathcal{HC}$, with associated data, is specified by four polynomial-time algorithms: setup, keygen, highncrypt and unhighncrypt.

setup: is a PPT algorithm that takes the security parameter κ as input and outputs the system parameter *params* to be used in the scheme. We assume the security parameter is always (maybe implicitly) encoded in *params*.

key-gen: is a PPT algorithm that takes the system parameter *params* as input and outputs a public-private key pair (pk, sk) used for highncryption and unhighncryption. For presentation simplicity, we assume *params* is included in pk. *In this work, we assume each user has a single key pair* (pk, sk)*, which is used both for highncryption and for unhighncryption.*

highncrypt: is a PPT algorithm that takes, as input, a sender's private key sk_s, the sender's public identity information $pid_s = (id_s, pk_s, cert_s)$ where $cert_s$ is sender's certificate issued by a certificate authority (CA), a receiver's public identity information $pid_r = (id_r, pk_r, cert_r)$, message $M \in \{0,1\}^*$ and associated data $H \in \{0,1\}^*$ to be highncrypted. It returns a highncryptext C, or symbol $\perp$ indicating highncryption failure. The associated data H, if any, appears in clear in the highncryptext $C \neq \perp$. *In this work, we allow a user to highcrypt a message to itself; that is,* $pid_s = (id_s, pk_s, cert_s)$ *may be equal to* $pid_r = (id_r, pk_r, cert_r)$. Also, we assume that some *offline-computable* intermediate randomness, e.g., DH-exponents, used in generating the highncryptest C is specified and stored in a variable $\mathcal{ST}_C$ that could be exposed to allow for a more robust security definition.

unhighncrypt: is a deterministic polynomial-time algorithm that takes, as input, a receiver's private key sk_r, the receiver's public identity information $pid_r = (id_r, pk_r, cert_r)$, and a highncryptext C. It outputs either (pid_s, M) or an error symbol $\perp$. *Note that, unlike in traditional unsigncryption, unhighncrypt does not take sender's public identity information* pid_s *as input.*

The *correctness* for a highncryption scheme requires that, for all sufficiently large security parameter κ, any key pairs (pk_s, sk_s) and (pk_r, sk_r) output by $\mathsf{key\text{-}gen}(1^\kappa)$, it holds $\mathsf{unhighncrypt}(sk_r, pid_r, \mathsf{highncrypt}(sk_s, pid_s, pid_r, H, M)) = (pid_s, m)$ for any $H, M \in \{0,1\}^*$ such that $\mathsf{highncryption}(sk_s, pid_s, pid_r, H, M) \neq \perp$.

We now present the *strong* security model for highncryption in the multi-user setting, where each user possesses a single key pair for both highncryption and unhighncrypton and can highncrypt messages to itself, and the adversary is allowed to adaptively register (dishonest) users.

Let n be the number of users in the system, where n is polynomial in the security parameter κ. The key pairs of all the honest parties in the system are generated

by the challenger according to the specified key generation algorithm. The adversary is given the public keys of all the honest users initially, *and can register arbitrary public keys (for dishonest parties) on its own.* Denote by HONEST (reps., DISHONEST) the set of public identity information for all the honest (resp., dishonest) parties in the system. Throughout this work, denote by pid_i, $1 \leq i \leq n$, the pubic identity information of user id_i, and by pid_s (resp., pid_r) the public identity information of the sender (resp., the receiver). The adversary is also given access to HO, UHO, EXO and Corrupt oracles, as specified below.

HO: On input (pid_s, pid_r, H, M) where $pid_r \in \mathsf{HONEST} \bigcup \mathsf{DISHONEST}$ and pid_r may be equal to pid_s, $H, M \in \{0,1\}^*$, HO returns $\mathsf{higncrypt}(sk_s, pid_s, pid_r, H, M))$ if $pid_s \in \mathsf{HONEST}$, otherwise, returns $\perp$. HO also stores, in private, some specified offline-computable intermediate randomness (in generating C) into $\mathcal{ST}_C$ in order to allow for later EXO query against C.

UHO: On input (pid_r, C), UHO returns $\mathsf{unhigncrypt}(sk_r, pid_r, C))$ if $pid_r \in \mathsf{HONEST}$, otherwise, returns $\perp$.

EXO (exposure oracle): On input $C \neq \perp$, EXO returns the value stored in $\mathcal{ST}_C$, i.e., the offline-computable intermediate randomness used in generating C, if C was output by an earlier HO query. Otherwise, $\perp$ is returned. This renders the adversary additional power, in contrast to traditional security definition of signcrypiton, and reflects the reality of bad randomness, various side-channel attacks and deployment in hostile environments (plagued with spyware or virus) where offline-computable values are more vulnerable to adversarial exposure.

Corrupt: On input $pid_i \in \mathsf{HONEST}$, $1 \leq i \leq n$, this oracle returns the private key sk_i of user id_i.

Outsider unforgeability. Informally, the goal of an outsider unforgeability adversary $\mathcal{A}^{OU}$ against $\mathcal{HC}$ is to forge a valid higncryptext created by an *uncorrupted* honest user pid_{s^*} for another *uncorrupted* honest user pid_{r^*}, where pid_{s^*} may be equal to pid_{r^*}, $1 \leq r^*, s^* \leq n$. Toward this goal, $\mathcal{A}^{OU}$ is allowed to issue HO, UHO, EXO and Corrupt queries. At the end of its execution, $\mathcal{A}^{OU}$ outputs (pid_{r^*}, C^*) as its forgery, where $pid_{r^*} \in \mathsf{HONEST}$ and the associated data contained in C^* in clear is denoted H^*. The advantage of $\mathcal{A}^{OU}$ for breaking outsider unforgeability, denoted $Adv_{\mathcal{A}^{OU},\mathcal{HC}}$, is defined to be the probability of the following conditions hold:

- $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*) = (pid_{s^*}, M^*)$, where $pid_{s^*} \in \mathsf{HONEST}$.
- $\mathcal{A}^{OU}$ has not issued $\mathsf{Corrupt}(pid_{s^*})$ query or $\mathsf{Corrupt}(pid_{r^*})$ query. But $\mathcal{A}^{OU}$ is allowed to query $\mathsf{EXO}(C^*)$ to expose the intermediate randomness used in generating C^*.
- C^* was not the output of $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$ issued by $\mathcal{A}^{OU}$. But $\mathcal{A}^{OU}$ is still allowed to query $\mathsf{HO}(pid_{s'}, pid_{r'}, H', M')$ for $(pid_{s'}, pid_{r'}, H', M') \neq (pid_{s^*}, pid_{r^*}, H^*, M^*)$,

in particular $(pid_{s^*}, pid_{r^*}, H', M^*)$ where $H' \neq H^*$, and can even query $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$ as long as the output returned is not equal to C^*. Also, parts of C^* (e.g., H^*) may appear in previous outputs of HO.

In traditional definitions of unforgeability for signcryption, $\mathcal{A}^{OU}$ is required to output $(pid_{s^*}, pid_{r^*}, M^*, C^*)$ as its forgery at the end of its execution, which implies that it "*knows*" the victim user pid_{s^*} and the message M^* being signcrypted. By comparison, our formulation does not make such a requirement. That is, $\mathcal{A}^{OU}$ may know neither pid_{s^*} nor M^*, even if (pid_{r^*}, C^*) is a valid forgery. Our security definition allows the exposure of $\mathcal{ST}_{C^*}$, i.e., the intermediate randomness used for generating the target higncryptext C^*, which is also not allowed in traditional security definitions of signcryption. In addition, security of associated data was not considered in traditional security definitions of signcryption, while strong unforgeability for the associated data is ensured by our definition. Consequently, our unforgeability formulation provides much more comprehensive and stronger security guarantee.

A higncryption scheme $\mathcal{HC}$ has *outside unforgeability*, if for any PPT adversary $\mathcal{A}^{OU}$ its advantage $Adv_{\mathcal{A}^{OU},\mathcal{HC}}$ is negligible for all sufficiently large security parameters. The definition of *inside unforgeability* is identical to that of outside unforgeability, except that oracle query $\mathsf{Corrupt}(pid_{r^*})$ is allowed to the adversary.

Insider confidentiality. Informally, the goal of an insider confidentiality adversary $\mathcal{A}^{IC}$ is to break the confidentiality of the message as well as *the public identity information* higncrypted to an *uncorrupted* honest target receiver by any (possibly *corrupted*) honest sender, even if $\mathcal{A}^{IC}$ is allowed to corrupt the sender and to expose the intermediate randomness used for generating other highcyptexts. For presentation simplicity, throughout this work we assume that all the users in the system have public identity information of equal length. But our security model and protocol constructions can be extended to the general case of different lengths of identities, by incorporating length-hiding authenticated encryption in the underlying security model and protocol constructions.

- *Phase 1:* $\mathcal{A}^{IC}$ is allowed to issue HO, UHO, EXO and $\mathsf{Corrupt}$ queries.
- *Challenge:* At the end of phase 1, $\mathcal{A}^{IC}$ outputs two equal length messages (M_0, M_1), an associated data H^*, and two pairs of public identity information of equal length $(pid_{s_0^*}, pid_{r^*})$ and $(pid_{s_1^*}, pid_{r^*})$ where $pid_{s_0^*}, pid_{s_1^*}, pid_{r^*} \in \mathsf{HONEST}$, and submits them to the challenger. The challenger chooses a random bit $\sigma \leftarrow \{0,1\}$, and gives $\mathcal{A}^{IC}$ the challenge higncryptext $C^* = higncrypt(sk_{s_\sigma^*}, pid_{s_\sigma^*}, pid_{r^*}, H$
- *Phase 2:* $\mathcal{A}^{IC}$ can continue executing as in phase 1, except asking $\mathsf{UHO}(pid_{r^*}, C^*)$ or $\mathsf{EXO}(C^*)$ or $\mathsf{Corrupt}(pid_{r^*})$ that will cause $\mathcal{A}^{IC}$ to trivially win the game. *But* $\mathcal{A}^{IC}$ *is allowed to issue* $\mathsf{Corrupt}(pid_{s_0^*})$ *and* $\mathsf{Corrupt}(pid_{s_1^*})$*, which captures forward ID-privacy.*

– *Guess:* Finally, $\mathcal{A}^{IC}$ outputs a bit σ'. $\mathcal{A}^{IC}$ wins the game, if $\sigma' = \sigma$.

A higncryption scheme $\mathcal{HC}$ has insider confidentiality, if for any PPT adversary $\mathcal{A}^{IC}$ and all sufficiently large security parameters, the adversary's advantage defined below is negligible:

$$Adv_{\mathcal{A}^{IC},\mathcal{HC}} = |2 \cdot Pr[\sigma' = \sigma] - 1| .$$

In the above security definition, if it is required to hold $pid^*_{s_0} = pid^*_{s_1}$ it is degenerated to the traditional insider confidentiality of signcryption, which is referred to as *insider message confidentiality* here. The definition of outside confidentiality is identical to that of inside confidentiality, except that neither $\mathsf{Corrupt}(pid^*_{s_0})$ nor $\mathsf{Corrupt}(pid^*_{s_1})$ is allowed.

5 Protocol Construction of Higncryption

In this section, we present a practical and conceptually simple (yet carefully designed) scheme integrating both higncryption and one-pass identity-Concealed Authenticated Key-Exchange (CAKE, for short), which consists of the following four algorithms, setup, keygen, $\mathsf{higncrypt}$ and $\mathsf{unhigncrypt}$.

SETUP. On a security parameter κ, $\mathsf{setup}(1^\kappa)$ returns $params = (G', N, G, g, q)$ specifying the underlying group over which the GDH assumption holds (as defined in Section 2).

KEY GENERATION. On the parameters $params$, for each honest user $i, 1 \leq i \leq n$, keygen takes $x_i \leftarrow Z_q^*$, sets $pk_i = g^{x_i} \in G$ and $sk_i = x_i$, and outputs the key-pair (pk_i, sk_i). The binding between user identity id_i and its public-key pk_i is authenticated by a certificate $cert_i$ issued by CA. Throughout this paper, unless otherwise stated, we assume that CA does not mandate proof-of-possession or proof-of-knowledge (POP/POK) of secret key during public key registration, but it performs sub-group membership check for each registered public key, i.e., checking $pk_i \in G \setminus 1_G$.[2]

HIGNCRYPTION. Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be an AEAD scheme, where $\mathcal{K}$ is the key space of K_{se}, $h : \{0,1\}^* \rightarrow \{0,1\}^l \cap Z_q^*$ be a cryptographic hash function where $l = \lceil |q|/2 \rceil$, $M \in \{0,1\}^*$ be the message to be higncrypted with associated data H, and $KDF : G \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a key derivation function. For presentation simplicity, we denote by Alice the sender who possesses public identity information $pid_A = (id_A, pk_A = A = g^a \in G, cert_A)$ and secret-key $sk_A = a \leftarrow Z_q^*$, and by Bob the receiver who possesses public identity information $pid_B = (id_B, pk_B = B = g^b \in G, cert_B)$ and secret-key $sk_B = b \leftarrow Z_q^*$. $\mathsf{higncrypt}(sk_A, pid_A, pid_B, H, M)$ works as follows:

[2] The subgroup test can be waived, if oracle access to EXO is denied in the security model of higncryption.

- Take $x \leftarrow Z_q^*$, and compute $X = g^x \in G$, $d = h(X, pid_A, pid_B)$, and $X' = X^d$. *The DH-exponent x can be generated offline, and is specified to be stored into $\mathcal{ST}_C$ that may suffer from adversarial exposure.*
- Compute $\overline{X} = AX' = AX^d$,[3] and pre-shared secrecy $PS = CDH(\overline{X}, B) = B^{a+xd} \in G$.
- Derive keys $(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$,[4] where $K_1 \in \mathcal{K}$, K_2 is empty for higncryption, or $K_2 \in \mathcal{K}$ for one-pass CAKE and in this case the joint distribution of (K_1, K_2) is computationally indistinguishable from uniform distribution over $\mathcal{K} \times \mathcal{K}$.
- Compute $C_{AE} \leftarrow \mathsf{Enc}_{K_1}(H, pid_A||X||M)$. Notice that the DH-component X is sent being encrypted.
- Finally, send the higncryptext $C = (H, \overline{X}, C_{AE})$ to the receiver.

UNHIGNCRYPTION. After receiving $C = (H, \overline{X}, C_{AE})$, $\mathsf{unhigncrypt}(sk_B = b, pid_B, C)$ works as follows:

- Compute the pre-shared secrecy $PS = CDH(B, \overline{X}) = \overline{X}^b \in G$, and derive the keys $(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$.
- Run $\mathsf{Dec}_{K_1}(H, C_{AE})$. If $\mathsf{Dec}_{K_1}(H, C_{AE})$ returns $\perp$, abort; otherwise, get $\{pid_A = (id_A, A, cert_A), X, M)\}$.
- Compute $d = h(X, pid_A, pid_B)$. If $\overline{X} = AX^d$ and pid_A is valid, accept (pid_A, M); otherwise, abort.

This integrated scheme of higncryption and one-pass CAKE is also presented in Fig. 1 (page 18). Below, we make some clarifications about the construction.

Note on subgroup test of $\overline{X}$. In the above protocol description, we have assumed the receiver checks that $\overline{X}$ is in the subgroup G of order q in G'. The basic technique for performing such a subgroup test is to verify $\overline{X} \in G' \backslash 1_G$ and $\overline{X}^q = 1_G$. However, if the cofactor t is small, e.g., $G' = Z_p^*$ such that $p = 2q + 1$ is a prime, or G is the subgroup of an elliptic curve over a finite field where the cofactor t is typically a small constant or just 1, the subgroup test of $\overline{X}$ can be essentially reduced to check $\overline{X} \in G'$ and $\overline{X}^t \neq 1_G$, which guarantees $\overline{X}$ is not in a small subgroup of G' of the order being a factor of t (though it may not fully ensure $\overline{X} \in G$). In practice, we recommend the following protocol variant with embedded subgroup test, where $PS = B^{t(a+xd)} = \overline{X}^{tb}$ and the receiver will abort if $\overline{X} \notin G'$ or $PS = 1_G$. We note that subgroup test can be waived, if the EXO queries are disallowed in security definition.

[3] An alternative way is to set $\overline{X} = A^d X$, which does not sacrifice provable security. We prefer to setting $\overline{X} = AX^d$, for the reason that, as we shall see in Sectioin 7, it allows more flexible and efficient implementations.

[4] Other ways to set the session-key K_2 for one-pass CAKE: $K_2 = KDF(PS, \overline{X}||pid_A||pid_B)$ or $K_2 = KDF(PS, X||pid_A||pid_B)$.

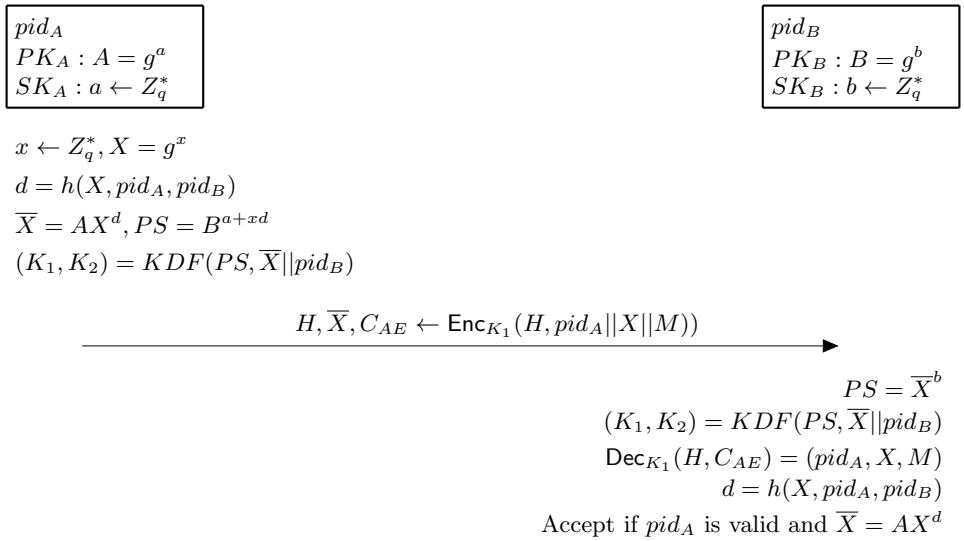

Fig. 1: Protocol structure of Higncryption

On the computation of d. In order to prevent or mitigate replay attack (which is nevertheless inevitable for any signcryption or one-pass AKE), a solution is to put a time-stamp t_A into the input of d, i.e., $d = h(X, pid_A, pid_B, t_A)$, and let t_A be a part of the associated data H or the message M. In addition, it appears that putting the message M into the input of d could possibly relax the security requirement on the underlying symmetric encryption used; but we wouldn't prefer to such a variant on the following grounds: (1) it may damage the modularity and independence of computing $\overline{X}$ from computing C, e.g., the offline pre-computability of $\overline{X}$ without knowing M; (2) AEAD already exists in most systems for secure communications.

One-pass CAKE. It is shown in [21,20] that signcryption implies one-pass AKE by setting the message M just to be the random session-key. But the session-key derived this way is dependent of the key generated for signcryption. When casting our higncryption scheme into one-pass identity-concealed authenticated key-exchange (CAKE), we set the session-key to be K_2 that is computationally independent of the key K_1 used for higncryption; that is, the exposure of K_1 does not affect the session-key security.[5] The analysis in [21,20] can be straightforwardly extended to show that this is a secure one-pass AKE. Compared to one-pass HMQV (HOMQV) [21], all the security properties of HOMQV remain with our higncrypiton-based one-pass CAKE, but our one-pass CAKE provides identity concealment.

[5] Another variant is to set the session-key of one-pass CAKE to be $KDF(K_1, X||pid_A||pid_B)$ or $HKDF - Extend(K_1, X||pid_A||pid_B)$. This provides extra security guarantee, as X is exchanged in the encrypted form.

Flexible implementation. Our higncryption scheme allows much flexible implementations, according to priorities and tradeoffs among security and efficiency in different application scenarios. Let $\mathcal{X}$ be a well-spread distribution over some subsets of of Z_q^* with min-entropy $\lambda_{\mathcal{X}} > \omega(\log |q|)$. For flexible implementations of higncryption, the only modification is: $X = g^x$ where x is taken according to $\mathcal{X}$ (rather than Z_q^*). As we shall show, provable security still holds with such a flexible implementation of higncryption.

In practice, at one's own discretion according to task criticality and application scenarios, the sender can take $x \leftarrow \{0,1\}^{\lceil |q|/4 \rceil} \cap Z_q^*$ (referred to as *light*-higncryption), or $x \leftarrow \{0,1\}^{\lceil |q|/2 \rceil} \cap Z_q^*$ (referred to as *medium*-higncryption), or just $x \leftarrow Z_q^*$ (referred to as *full*-higncryption). For example, if $|q| = 512$, we suggest $|x| = 128 = |q|/4$, i.e., *light*-higncryption, may suffice for many applications, e.g., those based on low-power devices like smart-phones where efficiency takes priority over security, or those based on devices with a trusted module to protect sender's static secret-key. In this case, the sender only performs about 1.75 exponentiations, compared to 2.5 (resp., 2) exponentiations in *full*- (resp., *medium*-) higncryption. For applications based on low-power devices like smart-phones, we may recommend to use *medium*-higncryption, where $x \leftarrow \{0,1\}^{\lceil |q|/2 \rceil} \cap Z_q^*$. Note that the security of medium-higncrption, even if sender's static secret-key $sk_A = a$ is exposed to adversary, is not reduced to solving half a DL-problem. Actually, computing X, letting along the exponent x, from $X' = X^{h(X)}$ seems already to be hard.

6 Security Proof of Higncryption

Assume KDF be a pseudorandom function with the following RO property, to get a value $KDF(PS, \varpi)$ where $PS \in G$ and $\varpi \in \{0,1\}^*$ the only way is to query the oracle KDF with input (PS, ϖ). For presentation simplicity, in the security analysis, we simply assume KDF to be a random oracle. The security analysis can be straightforwardly extended to the case that the output of the KDF oracle is pseudorandom (i.e., KDF is an RO with pseudorandom output). We assume the hash function h also to be a random oracle. As we concentrate on the security of the higncryption scheme, for presentation simplicity, for now we assume $K_1 = KDF(PS, \overline{X}||pid_B)$ (i.e., the session-key K_2 for one-pass CAKE is set to be empty). Throughout this work, for presentation simplicity, we also write $CDH(U, \cdot)$ simply as $CDH(pid, \cdot)$ when U denotes the public-key of user pid.

Theorem 4. *The higncryption scheme presented in Fig. 1 satisfies outsider unforgeability and insider confidentiality in the random oracle model, under the AEAD security and the GDH assumption.*

Security proof of outsider unforgeability Suppose that, after a series of adaptive oracle queries to HO, UHO, EXO and $\mathsf{Corrupt}$, the forger $\mathcal{A}^{OU}$ outputs, with non-negligible probability, a tuple (pid_{r^*}, C^*) where C^* contains in clear the associated data H^*, satisfying:

- $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*) = (pid_{s^*}, M^*)$, but C^* was not the output of $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$ issued by $\mathcal{A}^{OU}$. Note that $\mathcal{A}^{OU}$ is still allowed to query $\mathsf{HO}(pid_{s'}, pid_{r'}, H', M')$ for $(pid_{s'}, pid_{r'}, H', M') \neq (pid_{s^*}, pid_{r^*}, H^*, M^*)$, and can even query $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$ as long as the output returned is not equal to C^*.
- $pid_{s^*}, pid_{r^*} \in \mathsf{HONEST}$, and $\mathcal{A}^{OU}$ has not issued $\mathsf{Corrupt}(pid_{s^*})$ or $\mathsf{Corrupt}(pid_{r^*})$. But $\mathsf{EXO}(C^*)$ is allowed.

We assume the pair of honest users (pid_{s^*}, pid_{r^*}), for which successful forgery occurs with non-negligible probability, are fixed in advance, which can actually be correctly guessed with probability at least $\frac{1}{n^2}$ where n is the number of users in the system. We also assume that (pid_{r^*}, C^*) is the first successful forgery output by $\mathcal{A}^{OU}$; that is, it did not query $\mathsf{UHO}(pid_r, C)$ such that (pid_r, C) is also a valid forgery before outputting (pid_{r^*}, C^*). All these assumptions are only for presentation simplicity.

Given (pid_{s^*}, pid_{r^*}), the goal is to compute $CDH(pk_{s^*}, pk_{r^*})$, conditioned on that unforgeability is broken with non-nelgigible probability, where pk_{s^*} (resp., pk_{r^*}) is the public-key of the sender (resp., receiver) included in pid_{s^*} (resp., pid_{r^*}). We present the proof directly for the case of $pid_{s^*} = pid_{r^*}$, which is commonly viewed as the relatively harder case. The proof can be straightforwardly extended to the case of $pid_{s^*} \neq pid_{r^*}$.

Specifically, the simulator S takes $(params, pid_A)$ as input, where the public-key included in pid_A is denoted as $A = g^a$ for $a \leftarrow Z_q^*$ that is unknown to S, and its goal is to compute $CDH(A, A) = g^{a^2}$ with the help of a DDH oracle. Note that computing $CDH(A, A)$ is as hard as breaking the standard CDH assumption [32]. Toward this goal, S sets the public-key for user $s^* = r^*$ to be A, and sets the public and secret keys for all the other honest users in the system on its own. As a consequence, S can act on behalf of all the honest users except pid_A. Below we focus on the simulation of pid_A by S to deal with oracle queries made by the forger $\mathcal{A}^{OU}$ against pid_A.

We first note that S can perfectly handle all $\mathsf{Corrupt}$ queries allowed in the security game, where $\mathsf{Corrupt}(pid_{s^*})$ or $\mathsf{Corrupt}(pid_{r^*})$ is disallowed.

Consider a query $\mathsf{HO}(pid_s, pid_r, H, M)$, where pid_r is the public identity information of an arbitrary user in the system with public-key pk_r. First note that, if $pid_s \in \mathsf{DISHONEST}$, the output of $\mathsf{HO}(pid_s, pid_r, H, M)$ is simply defined to be "$\perp$". Also, if $pid_s \in \mathsf{HONEST}$ but $pid_s \neq pid_A$, this oracle query can be perfectly handled by the simulator itself. Hence, we only consider the case of $pid_s = pid_A$, i.e., $\mathsf{HO}(pid_A, pid_r, H, M)$. If $pid_r \in \mathsf{HONEST}$ but $pid_r \neq pid_A$, let $pk_r = B = g^b$

where $b \leftarrow Z_q^*$ is the secret-key actually set by the simulator itself. For this case, S works as the honest pid_A does, except that $PS = CDH(\overline{X}, B)$ is computed as $\overline{X}^b$. Otherwise (i.e., $pid_r \notin \mathsf{HONEST}$ or $pid_r = pid_A$), S computes $X = g^x$ and $\overline{X} = AX^d$, where $x \leftarrow Z_q^*$ and $d = h(X, pid_A, pid_r)$; S then sets K_1 to be a string taken uniformly at random from $\mathcal{K}$ of AEAD, computes $C_{AE} = \mathsf{Enc}_{K_1}(H, pid_A||X||M)$, and returns $C = (H, \overline{X}, C_{AE})$ as the output of $\mathsf{HO}(pid_A, pid_r, H, M)$. S also stores the tuple $(\overline{X}||pid_r, K_1)$ into a list $\mathcal{L}_{DDH}$ maintained by S itself and initiated to be empty. Note that in the later case, S cannot compute the pre-shared secrecy $PS = CDH(\overline{X}, pk_r)$ and consequently $KDF(PS, \overline{X}||pid_r)$. In order to keep the consistency of the random oracle KDF, from now on whenever the adversary $\mathcal{A}^{OU}$ makes an oracle query of the form $KDF(PS', \overline{X}||pid_r)$, based on the list $\mathcal{L}_{DDH}$ S checks whether $PS' = CDH(\overline{X}, pk_r)$ with the DDH oracle; if yes, it returns the pre-set value K_1.

In any case, the DH-exponent $x \leftarrow Z_q^*$ stored in $\mathcal{ST}_C$ is generated by the simulator S itself. As a consequence, S can perfectly handle all the EXO queries. So far, all the simulation for HO, $\mathsf{Corrupt}$ and EXO is perfect.

For a query $\mathsf{UHO}(pid_r, C = (H, \overline{X}, C_{AE}))$ made by $\mathcal{A}^{OU}$, we only consider the case of $pid_r \in \mathsf{HONEST}$ and $pid_r = pid_A$, as the rest cases can be perfectly handled by the simulator (note that, if $pid_r \in \mathsf{DISHONEST}$, UHO simply outputs "$\perp$"). S first checks whether C was ever output by $\mathsf{HO}(pid_s, pid_A, H, M)$ for some $M \in \{0,1\}^*$ and $pid_s \in \mathsf{HONEST}$, and outputs (pid_s, M) if so. Otherwise, for each KDF oracle query of the form $KDF(PS, \overline{X}||pid_A)$ made by $\mathcal{A}^{OU}$, the simulator checks whether $PS = CDH(\overline{X}, A)$ by the aid of the DDH oracle. If so, the simulator gets $K_1 = KDF(PS, \overline{X}||pid_A)$, uses K_1 to decrypt C_{AE}, and returns the result to $\mathcal{A}^{OU}$. Otherwise, S returns "$\perp$" indicating C is an invalid higncryptext (for user pid_r). Denote by "$\mathsf{failure}$" the event that, for some $(pid_A, C = (H, \overline{X}, C_{AE}))$ queried to UHO by $\mathcal{A}^{OU}$, the simulator outputs "$\perp$" while $\mathsf{UHO}(pid_A, C)$ does not. Conditioned on the "$\mathsf{failure}$" event does not occur, the simulation for UHO is perfect. Below, we show that the "$\mathsf{failure}$" event can occur with at most negligible probability.

Note that the $\mathsf{failure}$ event has already ruled out the possibility that C was the output of $\mathsf{HO}(pid_i, pid_A, H, M)$ for arbitrary $pid_i \in \mathsf{HONEST}$ and arbitrary (H, M). We now consider the possibility that $C = (H, \overline{X}, C_{AE})$ is the output of $\mathsf{HO}(pid_i, pid_j, H, M)$ made by $\mathcal{A}^{OU}$ for $pid_j \neq pid_A$ (and arbitrary pid_i, H, M). For this case, as $\overline{X}||pid_j \neq \overline{X}||pid_A$, it means that the shared-key generated by the random oracle KDF for computing C_{AE} and that for decrypting C_{AE} (when dealing with $\mathsf{UHO}(pid_A, C)$ in defining the "failure" event) are independent. By the security of AEAD as discussed in Section 2.1, $\mathsf{UHO}(pid_A, C)$ outputs $\perp$ with overwhelming probability. Thus, when the "$\mathsf{failure}$" event occurs w.r.t. $\mathsf{UHO}(pid_A, C = (H, \overline{X}, C_{AE}))$ where pid_A is the receiver, with overwhelming prob-

ability it holds: (1) C was not ever output by the HO oracle; (2) $\mathcal{A}^{OU}$ did not make the $KDF(PS, \overline{X}||pid_A)$ query for $PS = CDH(\overline{X}, A)$; and (3) (H, C_{AE}) make up a valid AEAD ciphertext w.r.t. $K_1 = KDF(CDH(\overline{X}, A), \overline{X}||pid_A)$.

We further consider two cases. (1) $K_1 = KDF(CDH(\overline{X}, A), \overline{X}||pid_A)$ was set by the simulator S when dealing with a query $\mathsf{HO}(pid_A, pid_A, H', M')$. In this case, S sets K_1 without querying the KDF oracle (but using its DDH oracle to ensure inconsistency of KDF). This implies that by the KDF security, with overwhelming probability, $\overline{X}$ is part of the output of $\mathsf{HO}(pid_A, pid_A, H', M')$ generated by the simulator. Denote by $(H', \overline{X}, C'_{AE})$ the output of the simulator S when dealing with $\mathsf{HO}(pid_A, pid_A, H', M')$. Note that (H', C'_{AE}) is the only AEAD ciphertext output by S w.r.t. K_1. As we assume $C = (H, \overline{X}, C_{AE}))$ was not ever output by the HO oracle, it means that $(H', C'_{AE}) \neq (H, C_{AE})$. This implies $\mathcal{A}^{OU}$ has output a new valid AEAD ciphertext (H', C'_{AE}) w.r.t. K_1, which can occur with negligible probability by the AEAD security. (2) Otherwise, with overwhelming probability, K_1 was neither set by S nor ever defined for the KDF oracle. In this case, also by the AEAD security, "$\mathsf{failure}$" occurs with negligible probability. We conclude that the $\mathsf{failure}$ event can occur with at most negligible probability, and consequently the view of $\mathcal{A}^{OU}$ in the simulation is computationally indistinguishable from that in the real attack game. Thus, successful forgery occurs also with non-negligible probability in the simulation.

Denote by $(pid_{r^*}, C^* = (H^*, \overline{X}^*, C^*_{AE}))$ the successful forgery output by $\mathcal{A}^{OU}$, satisfying $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*) = (pid_{s^*}, M^*)$ and C^* was not ever output by $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$, where pid_{s^*} and pid_{r^*} are the uncorrupted honest users that are assumed to have been correctly guessed by the simulator for presentation simplicity. Recall that we are considering $pid_{s^*} = pid_{r^*} = pid_A$.

For $(pid_{r^*}, C^* = (H^*, \overline{X}^*, C^*_{AE}))$ to be a successful forgery, $\mathcal{A}^{OU}$ must have made the RO query $h(X^*, pid_{s^*}, pid_{r^*}) = d^*$ such that $\overline{X}^* = pk_{s^*}X^{*d^*} = AX^{*d^*}$ that will be checked in computing $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*)$, where X^* may be generated by the adversary itself. Otherwise, $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*)$ returns $\perp$ with overwhelming probability in the random oracle model. Then, similar to the above argument for showing $\mathsf{failure}$ occurs with negligible probability in the UHO simulation, by the underlying AEAD security $\mathcal{A}^{OU}$ must have made the RO query to get $KDF(CDH(\overline{X}^*, pk_{r^*}), \overline{X}^*||pid_{r^*}) = KDF(CDH(\overline{X}^*, A), \overline{X}^*||pid_A) = K_1^*$.

The intuitive next idea is to rewind $\mathcal{A}^{OU}$ to the point that it just made the RO query $h(X^*, pid_{s^*}, pid_{r^*}) = h(X^*, pid_A, pid_A)$, and returns back a new random output $d^{*\prime}$. Then, by the general forking lemma [4], with non-negligible probability $\mathcal{A}^{OU}$ will also output a successful forgery $(pid_{r^*}, C^{*\prime} = (H^{*\prime}, \overline{X}^{*\prime}, C^{*\prime}_{AE}))$ in the second run after re-winding, where $\overline{X}^{*\prime} = AX^{*d^{*\prime}}$, and will make the query $KDF(CDH(\overline{X}^{*^{\prime}}, pk_{r^*}), \overline{X}^{*^{\prime}}||pid_{r^*}) = KDF(CDH(\overline{X}^{*^{\prime}}, A), \overline{X}^{*^{\prime}}||pid_A)$. It means that the simulator S can get $\alpha = CDH(\overline{X}^*, pk_{r^*}) = CDH(\overline{X}^*, A) = (AX^{*d^*})^a =$

$A^a X^{*d^* a}$, and $\beta = CDH(\overline{X}^{*\prime}, pk_{r^*}) = (AX^{*d^{*\prime}})^a = A^a X^{*d^{*\prime} a}$. From (α, β), S can compute $\gamma = CDH(X^*, pid_{r^*}) = CDH(X^*, A) = (\alpha/\beta)^{(d^* - d^{*\prime})^{-1}}$. Finally, S computes $CDH(A, A) = \alpha/(\gamma^{d^*})$, which violates the GDH assumption.

Unfortunately, a subtlety for correctly applying the forking lemma [4] is buried, and has been overlooked, in the above reasoning. The subtlety is specific to our higncryption construction, and its clarification might be of independent interest and be instrumental in analyzing future constructions of higncryption or other identity-hiding cryptographic schemes. Specifically, to apply the forking lemma, we need to ensure that the RO query $KDF(CDH(\overline{X}^*, pk_{r^*}), \overline{X}^* || pid_{r^*}) = KDF(CDH(\overline{X}^*, A), \overline{X}^* || pid_A)$ must be posterior to the RO query $d^* = h(X^*, pid_{s^*}, pid_{r^*}) = h(X^*, pid_A, pid_A)$.

Denote by $PS^* = CDH(\overline{X}^*, pk_{r^*}) = CDH(\overline{X}^*, A)$, where $\overline{X}^* = AX^{*d^*}$ is the value appeared in the successful forgery. In the random oracle model, there is only one approach for $\mathcal{A}^{OU}$ to make $KDF(PS^*, \overline{X}^* || pid_{r^*})$ prior to $h(X^*, pid_{s^*}, pid_{r^*})$. In more detail, suppose $(H, \overline{X}^*, C_{AE}) = higncrypt(pid_i, pid_{r^*}, H, M)$, where $pid_{r^*} = pid_A$, $pid_i \in \mathcal{DISHONEST}$ or $pid_i \in \mathcal{HONEST}$ but corrupted. Denote by $pk_i = C = g^c$, $\overline{X}^* = CX_i^{d_i} = Cg^{x_i d_i}$ where $d_i = h(X_i, pid_i, pid_{r^*})$. That is, the target $\overline{X}^*$ (appeared in the successful forgery) has already appeared in a former output of $higncrypt(pid_i, pid_{r^*}, H, M)$ for some $pid_i \neq pid_{s^*}$, satisfying $\overline{X}^* = CX_i^{d_i} = AX^{*d^*}$. Such an event is referred to as collision event. With this event, prior to the oracle query $h(X^*, pid_{s^*}, pid_{r^*}) = d^*$, the $KDF(PS^*, \overline{X}^* || pid_{r^*})$ oracle query was either made by the adversary $\mathcal{A}^{OU}$ itself or by the honest yet corrupted user pid_i. In either case, the adversary can compute $PS^* = CDH(\overline{X}^*, A)$, where $\overline{X}^*$ is either generated directly by $\mathcal{A}^{OU}$ (on behalf dishonest or corrupted user pid_i) or derived by corrupting pid_i and exposing x_i.

The observation here is that, in the RO model, the collision event can occur with at most negligible probability. Specifically, for any pair of $(pid_i, pid_j, X) \neq (pid_{i'}, pid_{j'}, X')$, the probability $\Pr[pk_i X^{h(X, pid_i, pid_j)} = pk_{i'} X'^{h(X', pid_{i'}, pid_{j'})}] \leq 2^{-l}$, where l is the output length of h. As the adversary $\mathcal{A}^{OU}$ is of polynomial-time t, the probability that $\mathcal{A}$ could cause the collision event to occur during its attack is negligible (specifically, at most $C_t^2 \cdot 2^{-l} < t^2 \cdot 2^{-(l+1)}$). This saves the applicability of the forking lemma to higncryption, and then finishes the proof of outsider unforgeability.

Security proof of insider confidentiality For presentation simplicity, we assume the challenger $\mathcal{C}$ has already correctly guessed the target receiver pid_{r^*}, which happens with probability $\frac{1}{n}$. The input of $\mathcal{C}$ is (B, X^*), where $B, X^* \leftarrow G \setminus 1_G$. Denote by $B = g^b$ and $X^* = g^{x^*}$, where $b, x^* \leftarrow Z_q^*$ that are however unknown to $\mathcal{C}$. The goal of the challenger $\mathcal{C}$ is to compute $CDH(B, X^*)$ with the aid of a DDH oracle. Towards this goal, $\mathcal{C}$ sets the public-key of the target receiver to be B, i.e., $pk_{r^*} = B$. $\mathcal{C}$ generates and sets the public/secret key pairs for all the rest

users in the system by itself, and will act on behalf of them. As a consequence, $\mathcal{C}$ can perfectly handle the oracle queries made by the adversary $\mathcal{A}^{IC}$ against all the users other than $pid_{r^*} = pid_B$. Similar to the proof of outsider unforgeability, $\mathcal{C}$ well simulates the target receiver pid_B with the aid of its DDH oracle.

When $\mathcal{A}^{IC}$ outputs two equal length messages (M_0, M_1), the associated data H and two pairs of public identity information of equal length $(pid^*_{s_0}, pid_{r^*})$ and $(pid_{s_1^*}, pid_{r^*})$, where $pid_{s_0^*}, pid_{s_1^*}, pid_{r^*} \in \mathsf{HONEST}$ and it is assumed that $pid_{r^*} = pid_B$, the challenger $\mathcal{C}$ chooses a random bit $\sigma \leftarrow \{0,1\}$ and sets the target higncryptext C^* as follows. For presentation simplicity, denote by $A = pk_{s_\sigma^*}$ the public-key of the user s^*_σ. Notice that it may be the case that $pid_{s^*_\sigma} = pid_{r^*}$ and thus $A = B$. $\mathcal{C}$ makes oracle query to get $d^* = h(X^*, pid_{s^*_\sigma}, pid_{r^*}) = h(X^*, pid_A, pid_B)$, where X^* is the input of $\mathcal{C}$, and computes $\overline{X}^* = AX^{*d^*}$. $\mathcal{C}$ checks whether the oracle query $KDF(CDH(\overline{X}^*, pk_{r^*}), \overline{X}^*||pid_{r^*}) = KDF(CDH(\overline{X}^*, B), \overline{X}^*||pid_B)$ has been made, with the aid of its DDH-oracle. If so, it outputs "$\mathsf{failure}$". Otherwise, it chooses K_1 uniformly at random from the key space $\mathcal{K}$ of AEAD, stores the tuple $(\overline{X}^*||pid_B, K_1)$ into the list $\mathcal{L}_{DDH}$, computes and returns $C^*_{AE} = \mathsf{Enc}_{K_1}(H, pid_{s^*_\sigma}||X^*||M_\sigma)$. From this point on, with the aid of its DDH oracle and the list $\mathcal{L}_{DDH}$, whenever $\mathcal{C}$ finds $\mathcal{A}^{IC}$ makes the query $KDF(CDH(\overline{X}^*, B), \overline{X}^*||pid_B)$ it just returns K_1 and records $CDH(\overline{X}^*, B)$.

First observe that, in the random oracle model, $X^{*d^*} = g^{x^* d^*}$ is distributed uniformly at random over $G \setminus 1_G$, where $x^* \leftarrow Z^*_q$ and h is assumed to be an RO. Consequently, $\overline{X}^* = AX^{*d^*}$ is distributed uniformly over $G \setminus 1_G$, even if $\mathcal{A}^{IC}$ knows $sk_{s_0^*}$ and $sk_{s_1^*}$ by user corruptions. This ensures that, on the one hand, $\mathcal{C}$ outputs "$\mathsf{failure}$" with negligible probability; and on the other hand, $\overline{X}^*$ perfectly hides the sender's identity information, even if both of $pid_{s_0^*}$ or $pid_{s_1^*}$ are corrupted. Then, by the AEAD security, to win the insider confidentiality game in the RO model, $\mathcal{A}^{IC}$ has to make the oracle query $KDF(CDH(\overline{X}^*, B), \overline{X}^*||pid_B)$ with non-negligible probability. *We remark that $\overline{X}^*$ being of distribution computationally indistinguishable from uniform distribution over $G \setminus 1_G$ suffices for the proof here.* For flexible implementations of higncryption where x is taken over a well-spread distribution with min-entropy greater than $\omega(\log |q|)$, by the flexible gap DL (FGDL) assumption introduced in Section 3, X^{*d^*} (and consequently $\overline{X}^*$) has distribution computationally indistinguishable from the uniform distribution over $G \setminus 1_G$ in the RO model.

Then, $\mathcal{C}$ rewinds $\mathcal{A}^{IC}$ to the point of making the oracle query $h(X^*, pid_A, pid_B)$, and redefines $d^{*\prime} = h(X^*, pid_A, pid_B)$, where $d^{*\prime}$ is taken uniformly at random from the range of h that is different from d^* with overwhelming probability, and re-runs $\mathcal{A}^{IC}$ from this rewinding point. By the forking lemma, with also non-negligible probability, $\mathcal{A}^{IC}$ will make the oracle query $KDF(CDH(\overline{X}^{*\prime}, B), \overline{X}^{*\prime}||pid_B)$ in the repeated run, where $\overline{X}^{*\prime} = AX^{*d^{*\prime}}$. From $\alpha = CDH(\overline{X}^*, B) = A^b X^{*bd^*}$ and

$\beta = CDH(\overline{X}^{*\prime}, B) = A^b X^{*bd^{*\prime}}$, $\mathcal{C}$ computes $CDH(X^*, B) = (\alpha/\beta)^{(d^*-d^{*\prime})^{-1}}$, which violates the GDH assumption.

7 Discussion and Applications of Higncryption

7.1 Brief Comparison

A brief comparison, among m-higncrypton (referring to medium higncryption), higncryption, (Zheng's) signcryption, and HOMQV, is given in Table 2. There, we only count the number of modular exponentiations (denoted "exp." in the table) for efficiency comparison.

		m-higncryption	higncryption	signcryption	HOMQV
efficiency	sender	2exp.	2.5 exp.	1 exp.	2 exp.
	receiver	1.5 exp.	1.5 exp.	2 exp.	1.5 exp.
forward-ID		√	√	X	X
x-security		√	√	X	√
receiver-deniability		√	√	X	√

Table 2: Comparison with Zheng's signcrypton and HOMQV

7.2 Application to QUIC

We note that higncryption is well compatible with the 0-RTT mode of QUIC, and can be easily implemented. Below, we first review the QUIC protocol according to the specifications given in [30]. Here, for presentation simplicity, the following protocol description does not fully coincide with (and omits many of) the technical details of QUIC.

QUIC supports two connection modes [30]: 1-RTT handles the case when the client tries to achieve a connection with a server for the first time in a particular time period. 0-RTT considers the case when the client is trying to connect to a server that it has already established at least one connection within that time period. In the initial connection within a time period, the server generates and sends to the client a state information stk, which is an AEAD encryption of the concatenation of the client's IP address $(\mathsf{IP}_C, \mathsf{IP}_S, \mathsf{port}_C, \mathsf{port}_S)$ and the time-stamp of the server ts_S. stk plays a role similar to that of the session ticket in TLS, which can be used by the client in later 0-RTT connection (as long as it does not expire and the client does not change its IP-address).

After getting stk and server's public identity information denoted pid_B here, the basic structure of QUIC with higncryption based 0-RTT connection is presented in Fig. 2. There, ts_C is client's time-stamp, k_{stk} is the AEAD key for generating stk

by the server. γ is an indicator variable newly introduced here, which is set to be 1 (resp., 0) for 0-RTT with (resp., without) client authentication. "$\{\ \}_K$" denotes AEAD encryption using key K, where the associated data contains the initial vector of AEAD, cid and packet sequence numbers. K_1 is derived from $CDH(\overline{X}, B)$ and some auxiliary information determined by the session transcript (including cid, pkt, nonc, stk, $\overline{X}$, pid_B, etc). The application key K_2 is derived from $CDH(\overline{X}, Y)$ and some auxiliary information (including cid, nonc, pkt, c_Y, etc). Note that, while K_1 does not provide perfect forward security (PFS) and the security against key compromising impersonation (KCI) attacks, the final application key K_2 does.

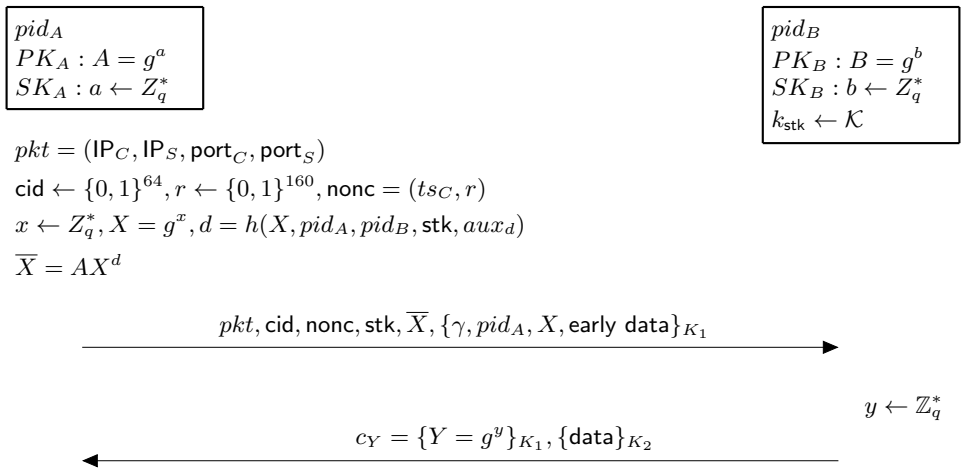

Fig. 2: Basic structure of QUIC with higncryption based 0-RTT connection

In order for providing more robust binding of $\overline{X}$ to the session it resides in and for preventing replay attacks, we set $d = h(X, pid_A, pid_B, \mathsf{stk}, aux_d)$, where $aux_d \in \{0,1\}^*$ is recommended to include cid and nonc. At the server side, it uses a mechanism, called the strike-register, to make sure that it does not process the same connection twice, by keeping track of used client's nonces within a limited amount of time in accordance with client's time-stamp ts_C. A server rejects a connection request from a client if its nonc is already included in its strike register or contains a time-stamp that is outside the allowed time range. Including client time-stamp ts_C in nonc also allows the server to detect clients whose clocks are too out-of-sync with the server (and hence vulnerable to expired certificates). For application scenarios where timing information may constitute a privacy concern, we may also suggest to get nonc, particularly ts_C, protected by the AEAD encryption, rather than being sent in clear.

7.3 Application to SACCE

Our higncryption also implies a two-round server-only authenticated and confidential channel establishment (SACCE) as defined in [27]. In this setting, only the server (corresponding to the sender in the higncryption scheme) possesses public/private key pair, and needs to authenticate itself to an anonymous client. This is the typical application scenario for TLS and QUIC. Our higncrypton can be adapted into a two-pass SACCE scheme, with the following modifications: (1) the public identity information of the receiver is replaced with a randomly generated DH-component; and (2) AEAD is replaced with a stateful length-hiding authenticated encryption (SLHAE) as defined in [27]. The higncryption based SACCE protocol is briefly described in Fig. 3, where $K = KDF(CDH(X, \overline{Y}), X||\overline{Y})$.

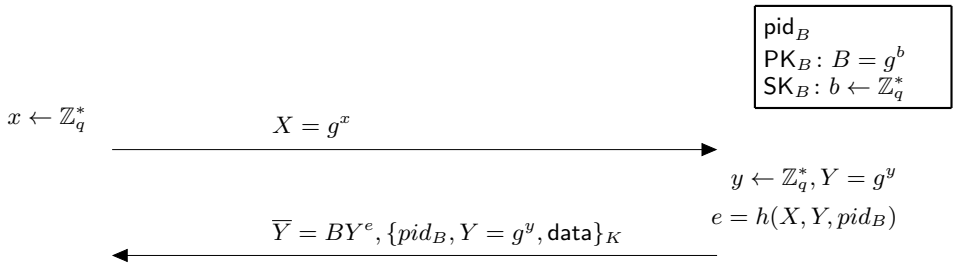

Fig. 3: Basic structure of higncryption-based SCAAE

We note that our higncrypton-based SCAAE is secure in accordance with the security definition given in [27], under the SLHAE security and the GDH assumption in the random oracle model. Here, we briefly highlight some key points for the SCAAE security proof.

- The session label $(X, \overline{Y})$ determines, actually commits to, the session-identifier (X, Y, pid_B). Specifically, given $\overline{Y} = BY^{h(X,Y,pid_B)}$ no PPT algorithm can output, with non-negligible probability, $(X', Y', pid_C) \neq (X, Y, pid_B)$ such that $\overline{Y} = BY^{h(X,Y,pid_B)} = CY'^{h(X',Y',pid_C)}$ in the random oracle model assuming h is an RO, where C is the public-key of pid_C.
- For server authentication, security is reduced to computing $CDH(X, B)$ with the aid of DDH-oracle (and under the SLHAE security), where X is the DH-component sent by the honest client in the test-session.
- For channel security, where server's static secret-key is allowed to be exposed for ensuring forward security, security is reduced to solving $CDH(X, Y)$, where X (resp., Y) is the DH-component generated in the test-session (resp., its matching session).

Compared to QUIC and TLS1.3 as described in Appendix A, the higncryption-based SCAAE is more efficient, signatureless, has forward ID-privacy and receiver deniability, strong resilience to exposure of intermediate state (whether y or $b+ye$, but not both of them), and enjoys flexible implementations and deployments.

7.4 Full TLS Handshake: The First Solution

Based on higncryption, DIKE [45,47] and QUIC, we present the first solution for full TLS handshake, which is described in Fig. 4. In subsequent sections, we will develop more efficient solutions.

In the protocol description, "$\{\ \}_K$" denotes AEAD encryption using key K, "$\{\ \}^*_K$" (resp., "$\{\ \}^+_K$") means it is optionally generated and sent only for 0-RTT (resp., 0.5-RTT) mode. SFIN (resp., CFIN) denotes server's (resp., client's) finish message as in TLS1.3. The key K_0 is derived from $CDH(\overline{X}, B)$ and auxiliary input (determined from public transcript and possibly pre-shared state). 1-RTT HS is a variable, which is set to be (pid_A, X) in the case of mutual authentication without 0-RTT, and is set to be empty in all the rest cases. The key K_1 and the session-key K are derived from $CDH(\overline{X}, Y))$ and auxiliary input, in a way that K_1 and K are computationally independent as is done in TLS1.3 or QUIC. The key K_2 is derived from both $CDH(\overline{X}, B)$ and $CDH(\overline{X}, Y)$ and auxiliary input.

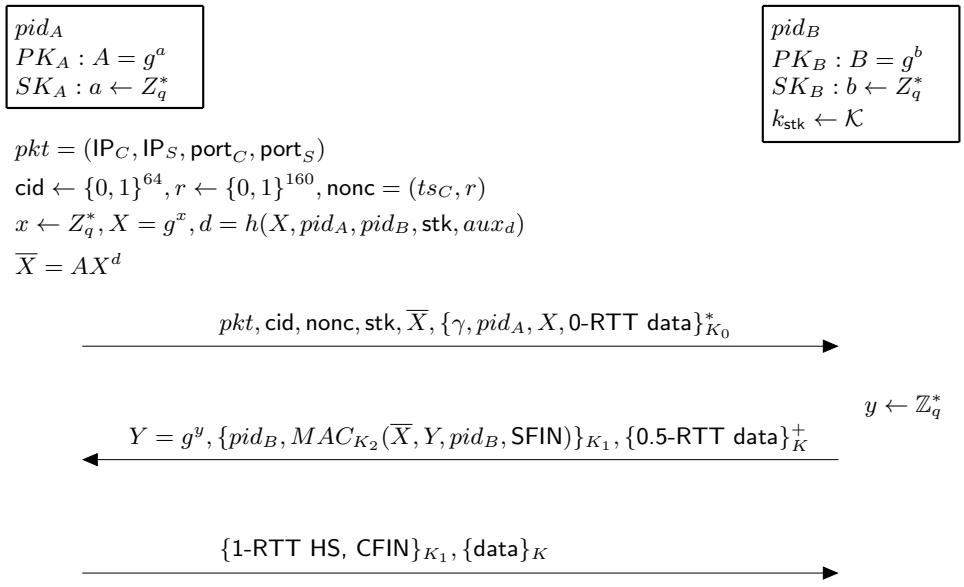

Fig. 4: First solution of TLS based on higncryption

For presentation simplicity, we have only presented 0-RTT with client authentication. Incorporating 0-RTT connection without client authentication is straight-

forward, where γ is set to "0", $\overline{X}$ is set to be $X = g^x$, and 1-RTT HS is set to be empty. For mutual authentication without 0-RTT, 1-RTT HS contains (X, pid_A); otherwise, 1-RTT HS is empty. For unilateral authentication mode, $\overline{X}$ is set to be $X = g^x$, and 1-RTT HS is set to be empty. We stress that, though some messages have different forms according to the modes of protocol run, however, all of these different modes of protocol run are *well unified within a single protocol structure in a modular and flexible way.*

8 A New Security Definitional Framework for Identity-Concealed Authenticated Key-Exchange (CAKE)

8.1 Motivation

Traditional CK-framework [11] and its variants (e.g., the post-ID CK-framework [12], eCK [29] and more) are not applicable to the analysis of CAKE. Session matching, which is at the heart of defining AKE security in the CK-framework, critically relies on the fact that players' identities and public-keys are exchanged in clear, and thus can be publicly verified merely according to session transcripts. Public verification of session matching is, in turn, crucial for provable composition with subsequent symmetric-key cryptographic primitives [11,7].

In the BR-framework [5], session matching is defined w.r.t. two sessions of identical session transcripts. Security according to the BR-framework requires that two sessions of different transcripts must have different session keys (otherwise, the security can be trivially broken). This causes limited applicability, or over strict security model and much more complex analysis, in order to deal with some issues as discussed below.

- An adversary could possibly cause two un-matching sessions: one complete session, and one incomplete session where the last message is simply dropped by the adversary, to be of the same session-key.
- For AKE protocols (like the TLS1.3 handshake) where some protocol messages are encrypted by a probabilistic symmetric encryption scheme using keys derived from some preliminary shared key (PSK), e.g., $PSK = CDH(X, Y)$, an adversary could get PSK in one session (e.g., by a state reveal query), and then uses PSK to re-encrypt plaintext messages of the exposed session in another session, which could also cause two sessions of different transcripts to have the same session-key.
- In practice, for AKE protocols deployed in a complex system, the protocol messages being exchanged may bear some non-critical values whose loss or modification may be tolerated in some application scenarios, which, however, cannot be permitted in the BR-framework.

Identity privacy for secure transport protocols, where the resultant session-key has already been used before session completion, was also formulated in some existing works (e.g., [8]), but being separated from the security definition for authenticated key-exchange or channel establishment.

Note that most secure transport protocols have (even intertwined) multiple stages. For example, the protocol run of QUIC within a time period is naturally divided into multiple stages: the original stage without 0-RTT connection, and subsequent stages with possible 0-RTT connections. For TLS1.3, its run can also be naturally divided into multiple stages: the original handshake, and subsequent resumption handshakes. A new security definitional framework for *multi-stage* key-exchange (MSKE) protocols is recently introduced in [18,15,7], and has proved its powerfulness in analyzing secure transport protocols like QUIC [18] and TLS1.3 [15]. Nevertheless, we notice that the MSKE framework is also not well applicable to identity-concealed authenticated key-exchange (CAKE), particularly for the classic deployment of AKE where key-exchange protocol is run only once at the beginning, i.e, in a single stage, and ceases as soon as the key is established and conveyed to the subsequent secure channel protocol.

Firstly, ID-privacy was not considered in the MSKE framework [18,15,7]. Secondly, to apply the MSKE framework to analyzing a CAKE protocol, we need to allow the adversary to expose the intermediate key, denoted k_{id}, used to encrypt player's identity information, as is done in [18,15,7]. This means that we have to divide the protocol run of the CAKE protocol (corresponding to a single stage in the standard deployment of AKE) into several stages, where the partial protocol run upon agreeing k_{id} should be treated as one stage [18,15]. In the MSKE framework, it is assumed that during the protocol run once a stage is finished, the protocol execution is immediately suspended and gives the control to adversary. This leads to several negative effects, as discussed below.

- The division of a single CAKE protocol run into multiple stages may be unnatural and even unrealistic. It may be the case that the messages sent by a player in one round (in a single flow or even in a single packet) may have to be divided into different stages. For example, with our CAKE protocol implementations, the DH-component and the accompanying AEAD ciphertext sent in the same round have to be divided into different stages: the DH-component belongs to an anterior stage which agrees on the key k_{id} (that will be exposed to adversary to argue the security of the subsequent stage), while the AEAD ciphertext belongs to the subsequent stage.
- It may result in unnatural modeling of secrecy exposure. Specifically, the intermediate key k_{id} is usually transient, particularly compared to some offline computable and stored values like DH-exponents and static secret-keys. If we allow the adversary the ability to expose so temporal secrecy like k_{id}, no secu-

rity guarantee can be made. On the other hand, if the ephemeral secrecy like k_{id} is allowed to be exposed but the exposure of long-lived states (like DH-exponents that can be offline pre-computed and stored) is denied, as is made in [18,15], it leads to unnatural or unrealistic modeling of secrecy exposure.

Thirdly, in this work we focus on definitional framework allowing a powerful concurrent man-in-the-middle (CMIM) adversary with adaptive party registration and strong capability of secrecy exposure. We note that in the formulation of the MSKE framework [18,15], the adversary isn't allowed to adaptive register users, has limited capability of secrecy exposure (where exposing DH-exponent is denied and static secret-key exposure isn't distinguished from user corruption), and is required to indicate the authentication type upon session initiation.

Thus, it would be much desirable to develop a definitional framework for identity-concealed authenticated key-exchange, which enjoys the following advantages simultaneously:

Composability. Session matching is publicly checkable, thus salvaging the composability.

Conciseness. Integration of both AKE security and identity privacy, thus simplifying security definition and analysis.

Unification. Unifying the dominant frameworks of CK, post-ID CK and BR.

Robustness and Versatility. Allowing a more powerful concurrent man-in-the-middle (CMIM) adversary with adaptive party registration and strong capability of secrecy exposure. And implying a list of important security properties in reality, like unknown key share (UKS), key compromise impersonation (KCI), concurrent non-malleability (CNM), perfect forward security (PFS), strong resilience to secrecy exposure, some of which are beyond the CK or BR framework.

8.2 System Setting

Suppose $\{U_1, \cdots, U_n\} = \mathsf{HONEST} \bigcup \mathsf{DISHONEST}$ are all the users in the system, where the public/secret key pairs for honest users in HONEST (resp., $\mathsf{DISHONEST}$) are set by the system (resp., the adversary itself), where the certificates for all the users are generated by a single certificate authority (CA). There is also a set $\mathsf{CORRUPTED} \subseteq \mathsf{HONEST}$ for indicating honest yet corrupted users. All the sets HONEST, $\mathsf{DISHONEST}$ and $\mathsf{CORRUPTED}$ may be initialized to be non-empty, and adaptively evolve during the attack. For presentation simplicity, n denotes the largest number of users in these sets. For a user U_i, $1 \leq i \leq n$, its public identity information $pid_i = (id_i, pk_i, cert_i)$, where id_i is U_i's identity, pk_i is its public-key and $cert_i$ is its certificate that contains a signature on (id_i, pk_i) made by the CA. Again, we assume the public identity information for all the users to

be of equal length (otherwise, we need length-hiding AEAD as underlying building tool in the protocol construction).

In the following description of adversarial setting, each session has a session-identifier *sid* that is simply assigned by an incremental counter. Setting session identifiers via a counter is only for distinguishing messages being delivered to different sessions, which is just an artefact in the security model. Each session, with session-identifier *sid*, keeps in private a local state $peer_{sid}$ (for indicating the interacting peer player), a local state ST_{sid} (for storing intermediate randomness) and a local state SK_{sid} (for storing session-key), all of which are originally initialized to be the empty string (meaning "undefined") and are assigned during the session run according to protocol specifications.

We say a session held at user U_i is complete or completed if U_i has successfully finished that session with resultant session-key, where U_i has sent or received the last message of that session. We say a session is incomplete (or on-going), if the session owner is still waiting for the next protocol message. We say a session is aborted, if it stops in the middle of session run because of some abnormal event (e.g., failure in authentication, etc) according to the protocol specifications. Whenever a session is aborted, all its local states are removed from memory. Whenever a session *sid* is completed, ST_{sid} is removed from memory but SK_{sid_I} is still kept in private. A session can also be expired, and for expired sessions the session-keys are also canceled.

8.3 Adversarial Setting

An adversary $\mathcal{A}$ against (a two-party) CAKE protocol is a concurrent man-in-the-middle (CMIM), who takes as input the security parameter and the public identity information for all the honest users in HONEST at the onset of its attack, and gets access to the following oracles:

Initiator: This oracle keeps a counter CTR_I that is initiated to be 0, and keeps in private a random bit $\sigma \leftarrow \{0,1\}$ (that is also embedded into the oracle Responder to be specified below).

- When receiving a special "(Start, U_i)" instruction, $1 \leq i \leq n$, for an *honest yet uncorrupted* user U_i (i.e., $U_i \in \mathsf{HONEST} \setminus \mathsf{CORRUPTED}$ upon receiving this query[6] (otherwise, it ignores the instruction), it sets $CTR_I := CRT_I + 1$, creates a session for the user U_i with session-identifier $sid_I = CTR_I$, and returns back $(sid_I, msg^{(1)}_{sid_I})$, where $msg^{(1)}_{sid_I}$ denotes the first protocol message for the session sid_I kept at Initiator for the initiator user U_i indicated by the adversary. It also creates, and keeps in private, a local state $peer_{sid_I}$ (for indicating the peer it

[6] But U_i may still be possibly corrupted after this query.

is interacting), a local state ST_{sid_I} (for storing intermediate randomness) and a local state SK_{sid_I} (for storing session-key) for session sid_I.
- Upon receiving an instruction of the form "(sid_I, msg_I^*)", it treats msg_I^* as the incoming message for session sid_I, and promptly responds back the next protocol message (according to protocol specification) if msg_I^* is not the last protocol message for that session; otherwise, it works according to the protocol specification.
- Upon receiving an instruction of the form "$(sid_I, \mathsf{ST\text{-}Exposure})$", it returns back the value stored in ST_{sid_I}, if session sid_I is not completed; otherwise, it ignores this instruction.
- Upon receiving "$(\mathsf{Test}, U_{t_0}, U_{t_1})$" for $U_{t_0}, U_{t_1} \in \mathsf{HONEST}$, where $1 \leq t_0 \neq t_1 \leq n$, it sets $U_t = U_{t_\sigma}$, and acts just as receiving the "(Start, U_t)" instruction, where the session-identifier set is denoted as sid_T for presentation simplicity.
- Upon receiving "$(sid_I, \mathsf{SK\text{-}exposure})$", it returns back the session-key SK_{sid_I}, if $sid_I \neq sid_T$ and sid_I has been completed yet not expired. If $sid_I = sid_T$, it returns SK_{sid_T} if $\sigma = 1$, otherwise it returns a value taken uniformly at random from $\{0,1\}^{sklen}$ where $sklen$ is the length of session-key.

Responder: This oracle keeps a counter CTR_R that is initiated to be 0, and embeds in private the same random bit σ used by Initiator.

- When receiving an instruction "$(\mathsf{Start}, U_j, msg_R^{(1)})$", $1 \leq j \leq n$, for an honest yet uncorrupted user U_j (otherwise, it ignores the instruction), it sets $CTR_R := CRT_R{+}1$, creates a session for the user U_j with session-identifier $sid_R = CTR_R$, treats $msg_R^{(1)}$ as the first-round incoming protocol message for session sid_R and returns back $(sid_R, msg_R^{(2)})$, where $msg_R^{(2)}$ denotes the second-round protocol message of session sid_R kept at Responder for the responder user U_j indicated by the adversary. It also creates, and keeps in private, a local state $peer_{sid_R}$, a local state ST_{sid_R}, and a local state SK_{sid_R} for session sid_R. We remark that the same user can be indicated to be both initiator (in one session held at Initiator) and responder (in another session held at Responder).
- Upon receiving an instruction "(sid_R, msg_R^*)", it treats msg_R^* as the incoming message for session sid_R, and responds back promptly the next protocol message if msg_R^* is not the last protocol message of that session.
- Upon receiving "$(sid_R, \mathsf{ST\text{-}Exposure})$" instruction, it returns back the value stored in ST_{sid_R}, if session sid_R is not completed; otherwise, it ignores this instruction.
- Upon receiving "$(\mathsf{Test}, U_{t_0}, U_{t_1}, msg_R^{(1)})$" for $U_{t_0}, U_{t_1} \in \mathsf{HONEST}$, where $1 \leq t_0 \neq t_1 \leq n$, it sets $U_t = U_{t_\sigma}$, and acts just as receiving the "$(\mathsf{Start}, U_t, msg_R^{(1)})$" instruction, where the session-identifier set is denoted as sid_T for presentation simplicity. We stress that the Test-type query can be made by the adversary

for only once during its attack, exclusively against Initiator or Responder(but not both of them).
- Upon receiving "$(sid_R, \mathsf{SK\text{-}exposure})$", it returns bask the session-key SK_{sid_R} if $sid_R \neq sid_T$ and the session sid_R has been completed yet not expired. If $sid_R = sid_T$, SK_{sid_T} is returned if $\sigma = 1$, otherwise a value taken uniformly at random from $\{0,1\}^{sklen}$ is returned.

Peer: Upon on an input "sid", it returns back the value stored in $peer_{sid}$, if sid is an existing session held at Initiator or Resonder (otherwise, the query is ignored).[7]

StaKey: Upon receiving "U_i", $1 \leq i \leq n$, this oracle returns back the static secret-key of U_i if $U_i \in \mathsf{HONEST}$; otherwise, it ignores the query.

Corrupt: Upon receiving "U_i", $1 \leq i \leq n$, if $U_i \in \mathsf{HONEST}$ (otherwise, the query is ignored) this oracle returns back all the information (including static secret-key, intermediate randomness, and session-keys, etc) in the memory part (maintained by Initiator and/or Responder) for U_i, and sets $\mathsf{CORRUPTED} = \mathsf{CORRUPTED} \bigcup \{U_i\}$. That is, we allow the adversary to adaptively corrupt honest users.

Register: Upon a query "(U_i, honest), where U_i is a new user not in $\mathsf{HONEST} \bigcup \mathsf{DISHONEST}$ when the query is made, it generates the public/secret key pair and gets the certificate for U_i (with the aid of CA), returns back the public identity information of U_i (including its identity, public-key and certificate), and sets $\mathsf{HONEST} = \mathsf{HONEST} \bigcup \{U_i\}$. Upon a query "$((U_j, pk_j), \mathsf{dishonest})$ for a new user U_j, it gets the certificate for (U_j, pk_j) (with the aid of CA), returns back the public identity information of U_j, and sets $\mathsf{DISHONEST} = \mathsf{DISHONEST} \bigcup \{U_j\}$. That is, we allow the adversary to adaptively register users in the system. Note that each user cannot have multiple certificates (with the same CA), but the adversary can register a dishonest user with the public-key of an honest user.

During its attack, the adversary $\mathcal{A}$ schedules all the oracle queries adaptively as it wishes. At end end of the attack, $\mathcal{A}$ outputs a bit σ'.

8.4 CAKE Security Definition

In our model, the label of a session is defined to be a substring of the session transcript. For example, the session label for CAKE in Fig. 7 is defined to be $(\overline{X}, \overline{Y})$. Two sessions are matching, if they have the same session label.

Let sid_T be the completed test-session held at the user $U_t = U_{t_\sigma}$ with $peer_{sid_T} = U_k \in \mathsf{HONEST}$, $1 \leq k \leq n$, where U_t may be identical to U_k. Denote by sid'_T its matching session (in case the matching session exists), which may be still on-going. We say the test-session is *exposed* during the attack, if any of the following events occurs:

[7] Note that the value returned may be an empty string, in case the peer of that session has not been determined.

- U_{t_0} or U_{t_1} is corrupted via the Corrupt query,[8] or ST_{sid_T} was exposed via the $(ST_{sid_T}, \mathsf{ST\text{-}exposure})$ query.[9]
- The static secret-key of U_k is exposed.[10]
- The query $(sid, \mathsf{SK\text{-}exposure})$, $sid = sid_T$ or $sid = sid'_T$ (in case the matching session exists), was issued.
- The query $\mathsf{Peer}(sid'_T)$ was issued.

Note that, for unexposed test-session, it may be the case: both the static secret-key of U_{t_0} and that of U_{t_1} are exposed, and $ST_{sid'_T}$ is exposed.

Definition 1 (Strong CAKE-security). *A two-party key-exchange protocol is* strongly *CAKE-secure, if for any PPT adversary $\mathcal{A}$ as defined above, and for any sufficiently large security parameter, it holds:*

Label-security: *Any of the following events occurs with negligible probability:*
- *There exist more than two sessions of the same session label.*
- *There exist two matching sessions: session sid held at user U_i and session sid′ held at user U_j, such that any of the following events occurs:*
  - *U_i and U_j play the same session role (i.e., both of them are initiators or responders).*
  - *$SK_{sid} \neq SK_{sid'}$.*
  - *$peer_{sid} \neq \perp \wedge peer_{sid} \neq U_j$, or $peer_{sid'} \neq \perp \wedge peer_{sid'} \neq U_i$.*[11]

  We remark that label-security is w.r.t. arbitrary PPT adversaries who can, in particular, expose the static secret-keys of all honest users and expose the local states of all existing sessions.

ID-concealed session-key (ICSK) security: *On condition that the test-session sid_T (held at the uncorrupted user U_t) is completed and unexposed, the following quantities all are negligible:*

Impersonation security: *The probability that the test-session has no matching session.*

ID-SK indistinguishability: $|\Pr[\sigma' = \sigma] - \frac{1}{2}|$. *Note that, this in particular implies both perfect forward security (PFS) and forward ID-privacy, as the static secret-keys of both U_{t_0} and U_{t_1} could be exposed.*

---

[8] However, exposing the static secret-key of U_{t_0} and/or that U_{t_1}, via the StaKey oracle, does not necessarily expose the test-session.

[9] If $U_{t_\sigma} = U_k$, i.e., the session holder and session peer are the same user for the test-session, ST_{sid_T} is allowed to be exposed.

[10] In case the matching session sid'_T exists, this can be relaxed that either static secret-key of U_k or $ST_{sid'_T}$ is unexposed.

[11] It implies that: (1) If both sid and sid' are complete, then $peer_{sid} = U_j \wedge peer_{sid'} = U_i$. (2) if both of these two matching sessions are incomplete, it could be $peer_{sid} = peer_{sid'} = \perp$. (3) if only one session (w.l.o.g., the session sid) is completed while the other session (say, sid') is incomplete, it could be: $peer_{sid} = U_j$ but $peer_{sid'} = \perp$. The last case models the asynchronism between defining $peer_{sid}$ and defining $peer_{sid'}$, or the unavoidable dropping message attacks by adversary.

Discussion. In a sense, the session label could be viewed as a *concurrent non-malleable* commitment to players' identities and session-key. In particular, label-security implies the security against unknown key share (UKS) attack. ICSK-security integrates both identity concealment and a strong AKE security. Note that, as the static secret-keys of both U_{t_0} and U_{t_1} could be exposed, impersonation security (resp., ID-SK indistinguishability) implies security against KCI attacks (resp., perfect forward security and forward ID-privacy). Entity authentication, implied by label security and impersonation security together, is very strong, which says that the CMIM adversary cannot impersonate an honest user in a session no matter whether the matching session exists or not. Label security, together with ID-SK indistinguishability, implies that: there are at most two sessions can be matching, and matching sessions have matched peer view and the same session-key, and unmatched sessions must have different (computatinally independent) session-keys.

The security definition in accordance with CK or BR framework can be viewed as a special case of our CAKE-security formulation. When being cased into the CK-framework (resp., BR-framework), the session label needs to include players' identity information (resp., the whole session transcript). Note also that, in the security model of CAKE, the adversary indicates the session holder when starting a session run, but does not necessarily indicate the peer player for the session. However, the session peer can be exposed via oracle queries to Peer. This way of formulation, on the one hand, allows a more powerful adversary, and on the other hand, incorporates the post-ID CK-framework [12] (where peer identity is not necessarily known at the start of session run) as a special case.

In general, we can treat ID-indistinguishability (ID-IND) and SK-indistinguishability (SK-IND) separately w.r.t. two test-sessions: one for defining ID-IND and one for SK-IND, by embedding a pair of independent random bits (rather than a single random bit σ). In this case, we can allow the adversary to have more powerful ability of secrecy exposure. For presentation simplicity, we prefer to the proposed more concise formulation.

8.5 Adaption to Unilateral CAKE (UCAKE)

The security definitional framework for CAKE protocols with mutual authentication can be adapted to CAKE protocols with unilateral authentication, which is referred to as unilateral CAKE (UCAKE). This is actually the most common application scenario of secure transport protocols like SSL/TLS and QUIC. For presentation simplicity, we assume only the responder (server) authenticates it to the initiator (client), while the client may not necessarily possess public iden-

tity information. [12] In this setting, for a session sid_R run at Responder, the local variable $peer_{sid_R}$ is always empty indicating "undetermined".

The adversarial setting is the same as that for CAKE with mutual authentication, except that: (1) the output of either $\mathsf{Corrupt}(U_i)$ or $\mathsf{StaKey}(U_i)$, for an initiator user U_i who does not possess public-key, includes a special symbol "$\perp$" in the place allocated for static secret-key. (2) We additionally allow the adversary to register honest or dishonest users without public identity information, where no CA gets involved. (3) For the pair of users (U_{t_0}, U_{t_1}) specified by the adversary in the test-query, we require that either both of them have public-keys or both of them do not; and if the test-query is against Responder they must both have public-keys.[13]

Note that the client may still possess public identity information, and we still allow the adversary to indicate the session holder when starting a session at Initiator. The reasons are as follows. Firstly, it is for presentation consistency with the definitional framework of CAKE security. Secondly, like in SSL/TLS, a client user usually does not know whether its authentication is required or not when starting the session run of handshake, which may be dependent upon the request of the server. Thirdly, for the sake of privacy protection, a client may want to be anonymous firstly with a UCAKE protocol run, and then reveals its public identity information (but protected with some key derived from the first UCAKE run) and makes authentication in a later stage, e.g., in the re-negotiation or resumption phase of SSL/TLS. In this sense, our presentation is concise yet better reflects the reality, gives more power to adversary, and enjoys ease of future model extensions.

Let sid_T be the completed test-session held at the user $U_t = U_{t_\sigma}$, where $peer_{sid_T} = U_k \in \mathsf{HONEST}$ (in case sid_T is run at Initiator) or $peer_{sid_T}$ is an empty string representing undefined (if sid_T is run at Responder). Denote by sid'_T its matching session held by a user U_j (in case the matching session exists), which may still be on-going. We say the test-session is *exposed* for UCAKE during the attack, if any of the following events occurs:

- U_{t_0} or U_{t_1} is corrupted via the Corrupt query, or ST_{sid_T} was exposed via the $(ST_{sid_T}, \mathsf{ST\text{-}exposure})$ query.[14]
- The query $(sid, \mathsf{SK\text{-}exposure})$, $sid = sid_T$ or $sid = sid'_T$, was issued.
- sid_T is run at Initiator but the static secret-key of U_k is exposed.[15]

[12] In general, a client can possess public identity information. In this case, it just does not use its identity information in the run of UCAKE, but may reveal its identity information when client authentication is required in a later stage, e.g., in a renegotiation or resumption session of TLS.

[13] Actually, if the test-query is against Initiator, specifying the pair of users (U_{t_0}, U_{t_1}) does not make sense for UCAKE. The treatment here is for presentation consistency with the definitional framework of CAKE.

[14] If we only require session-key indistinguishability, ST_{sid_T} can be exposed if sid_T is run at Responder.

[15] If the matching session sid'_T exists, it can be relaxed that either of secret-key of U_k or $ST_{sid'_T}$ is unexposed.

– sid_T is run at Responder but any of the following holds: the matching session sid'_T does not exist, or U_j (i.e., the session owner of sid'_T) is corrupted, or $ST_{sid'_T}$ is exposed, or the query $\mathsf{Peer}(sid'_T)$ was issued.

In other words, for the completed test-session sid_T run at Responder to be unexposed, the following conditions must hold simultaneously: (1) The matching session sid'_T exists at an uncorrupted honest client user U_j. (2) The adversary didn't make the query $\mathsf{Peer}(sid'_T)$. (3) Neither U_{t_0} nor U_{t_1} is corrupted, *but the static secret-keys of both* U_{t_0} *and* U_{t_1} *can be exposed* (which particularly implies perfect forward security and forward ID-privacy). (4) Neither ST_{sid_T} nor $ST_{sid'_T}$ is exposed. （5）Neither the session-key of sid_T nor that of sid'_T is exposed. For the test-session sid_T run at Initiator to be unexposed, the following must hold simultaneously: (1) The static secret-key of the peer user U_k is unexposed. (2) The local state of ST_{sid_T} is unexposed (*but the local state of* $\mathsf{ST}_{sid'_T}$ *of the matching session can be exposed*). (3) Neither U_{t_0} nor U_{t_1} is corrupted. (4) Neither the session-key of sid_T nor that of sid'_T is exposed.

Definition 2 (UCAKE-security). *A two-party key-exchange protocol is* strongly *UCAKE-secure, if for any PPT adversary* $\mathcal{A}$ *as defined above, and for any sufficiently large security parameter, it holds:*

Label-security: *Any of the following events occurs with negligible probability:*

- *There exist more than two sessions of the same session label.*
- *There exist two matching sessions: session sid held at user* U_i *and session* sid' *held at user* U_j, $1 \leq i, j \leq n$, *such that any of the following events occurs:*
  - U_i *and* U_j *play the same session role (i.e., both of them are initiators or responders).*
  - $SK_{sid} \neq SK_{sid'}$.
  - $peer_{sid} \neq \perp \bigwedge peer_{sid} \neq U_j$ *if* U_j *is the responder, or* $peer_{sid'} \neq \perp \bigwedge peer_{sid'} \neq U_i$ *if* U_i *is the responder.*

ID-concealed session-key (ICSK) security: *Supposing the test-session* sid_T *(held at the uncorrupted user* U_t*) is completed and unexposed, the following quantities all are negligible:*

Impersonation security: *The probability that the test-session* sid_T *has no matching session, on condition that* sid_T *is run at* Initiator.

ID-SK indistinguishability: $|\Pr[\sigma' = \sigma] - \frac{1}{2}|$.

9 Protocol Construction and Security Analysis of UCAKE

In this section, we present a two-round protocol for identity-concealed authenticated key-exchange with unilateral authentication (UCAKE), and prove its security in accordance with our strong UCAKE-security definiton (Definition 2).

9.1 Protocol Construction of UCAKE

Let (G', N, G, g, q) specify the underlying group over which the GDH assumption holds. For each honest user $i, 1 \leq i \leq n$, its secret-key sk_i is set to be $x_i \leftarrow Z_q^*$, and its public-key pk_i is set to be g^{x_i}. The binding between user identity id_i and its public-key pk_i is authenticated by a certificate $cert_i$ issued by a certificate authority (CA). Throughout this paper, unless otherwise stated, we assume that the CA does not mandate proof-of-possession or proof-of-knowledge (POP/POK, for short) of secret key during public key registration, but it performs sub-group membership check for each registered public key, i.e., checking $pk_i \in G \setminus 1_G$. [16] Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be an AEAD scheme, where $\mathcal{K} = \{0,1\}^\kappa$ is the key space of K_{se}, $h : \{0,1\}^* \rightarrow \{0,1\}^l \cap Z_q^*$ be a cryptographic hash function where $l = \lceil |q|/2 \rceil$, and $KDF : G \times \{0,1\}^* \rightarrow \{0,1\}^\kappa \times \{0,1\}^\kappa$ be a key derivation function. For presentation simplicity, we denote by Alice the anonymous client who does not necessarily possesse public identity information, and by Bob the server who possesses public identity information $pid_B = (id_B, pk_B = B = g^b, cert_B)$ and secret-key $sk_B = b \leftarrow Z_q^*$. The protocol structure of UCAKE is depicted in Fig. 5 (page 40), where H_B is the associated data for encrypting $pid_B||Y$.[17] For a session run, the local state ST_{sid} is specified to be the DH-exponent y (if sid is run at $\mathsf{Responder}$) or x (if sid is run at $\mathsf{Initiator}$), which can be offline computed and stored.

We assume the server always performs subgroup membership test for the incoming DH-exponent X explicitly or implicitly. The explicit subgroup test is to verify that $X \in G' \setminus 1_G$ and $X^q = 1_G$. The implicit subgroup test is to check $X \in G'$ and $X^t \neq 1_G$, which guarantees X is not in a small subgroup of G' of the order being a factor of t (though it may not fully ensure $X \in G$). In practice, we recommend the following protocol variant with implicit subgroup test, where $PS = X^{t(b+ye)}$ and the server will abort if $X \notin G'$ or $PS = 1_G$. Such an implicit subgroup test costs almost for free when G is the subgroup of an elliptic curve over a finite field, where the cofactor t is typically a small constant or just 1. We note that the subgroup test can be waived, if oracle query to EXO is denied in the UCAKE-security definition.

[16] The subgroup test can be waived, if no ephemeral secrecy is exposed.

[17] In reality, the associated data is usually implicitly determined from the context (e.g., the hash of the transcript of protocol run or some pre-determined states).

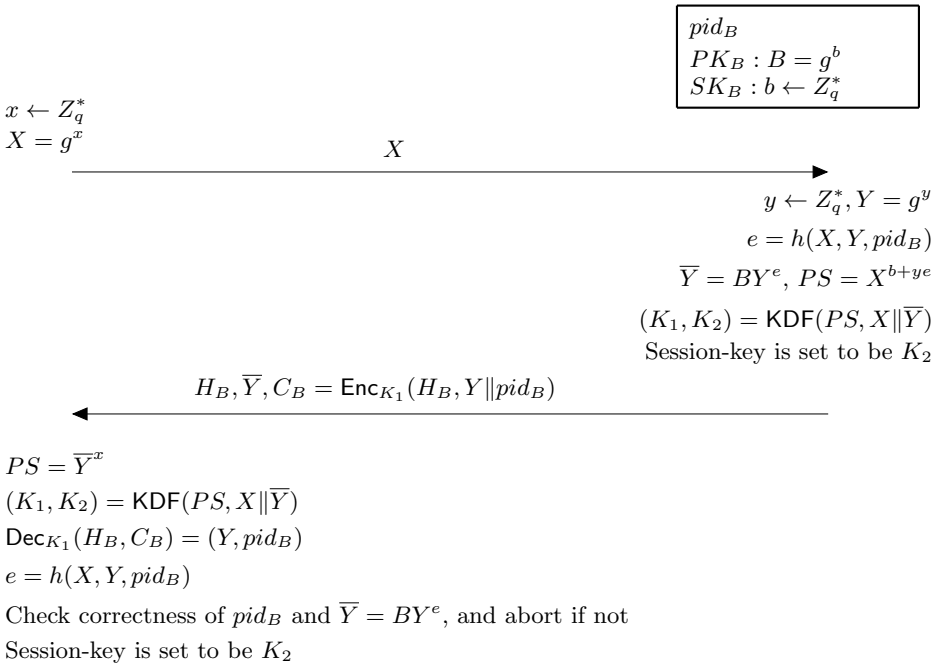

Fig. 5: Protocol structure of UCAKE

In the above protocol description, we used an abstract key-derivation function KDF. An alternative way to set the session-key K_2 is: $K_2 = KDF(PS, X||Y||pid_B)$. In practice, KDF can be instantiated with HKDF as in TLS1.3, or with the two-stage key derivation process via HMAC as specified by NIST SP800-56C. For such KDF instantiations, some random nonces and other auxiliary information may need to be exchanged, which are put into the input of KDF. In general, the AEAD ciphertext C_B can also encrypt some data directly, or as in 0.5-RTT mode of TLS1.3 the server uses the session-key K_2 to encrypt application data before the client confirms the session-key. The session-key could also be set to be K_1. In these cases, we need to use the ACCE security model.

9.2 Security Analysis of UCAKE

Theorem 5. *The protocol presented in Fig. 5 is strongly UCAKE-secure, under the AEAD security and the GDH assumption in the random oracle model.*

Proof. For a session run of the protocol described in Fig. 5, define its label to be $X||\overline{Y}$. We first prove the following lemma, which says that $\overline{Y}$ is actually a commitment to (X, Y, pid_B).

Lemma 3. *Assuming* $h : \{0,1\}^* \rightarrow \{0,1\}^l \cap Z_q^*$ *is an RO, where* $l = \lceil |q|/2 \rceil$, *no PPT algorithm can output* $\{aux \in \{0,1\}^*, Y \in G, pid_j = (id_j, pk_j, cert_j)\}$ *and* $\{aux' \in \{0,1\}^*, Y' \in G, pid_k = (id_k, pk_k, cert_k)\}$, *where* $\{aux, Y, pid_j\} \neq \{aux', Y', pid_k\}$ *and* $1 \leq j, k \leq n$, *such that* $pk_j Y^{h(aux,Y,pid_j)} = pk_k Y'^{h(aux',Y',pid_k)}$.

Proof (of Lemma 3). For any pair of different $\{aux, Y, pid_j\}$ and $\{aux', Y', pid_k\}$, the probability that $pk_j Y^{h(aux,Y,pid_j)} = pk_k Y'^{h(aux',Y',pid_k)}$ is $\frac{1}{2^l-1}$ assuming $h : \{0,1\}^* \rightarrow Z_q^*$ is a random oracle. Then, for any PPT algorithm who makes at most t oracle queries to h, where t is polynomial in $|q|$, the probability it outputs a pair of different $\{aux, Y, pid_j\}$ and $\{aux', Y', pid_k\}$ such that $pk_j Y^{h(aux,Y,pid_j)} = pk_k Y'^{h(aux',Y',pid_k)}$ is at most $\frac{t^2}{2^l-1}$, which is negligible. □

We first prove the label-security. It is straightforward that, with overwhelming probability, no more than two sessions (run at Initiator and Responder) can have the same label. Let session sid held at user U_i and session sid' held at user U_j, $1 \leq i, j \leq n$, are matching (i.e., they have the same session label $X||\overline{Y}$). Note that, for honestly generated X and $\overline{Y}$, with overwhelming probability $X \neq \overline{Y}$, which holds even if y is taken from a well-spread distribution over Z_q^* with min-entropy greater than $\omega(\log |q|)$. Label matching (with inequality between X and $\overline{Y}$) implies that U_i and U_j cannot play the same session role. As $PS = CDH(X, \overline{Y})$ and session-key is derived from $KDF(PS, X||\overline{Y})$, we conclude that the two matching sessions must have the same session-key. Finally, for presentation simplicity, suppose U_i is the initiator and U_j is the responder w.r.t. the two matching sessions sid and sid' of the same session label $X||\overline{Y}$, but $peer_{sid} \neq U_j$. Note that, in the session sid' held by the responder user U_j, it holds that $\overline{Y} = pk_j Y^{h(X,Y,pid_j)}$. The fact that $peer_{sid} \neq U_j$ at the initiator user U_i means that: there exist a PPT adversary who successfully opens $\overline{Y} = pk_j Y^{h(X,Y,pid_j)}$ into (X, Y', pid_k) in the session sid, where $pid_k \neq pid_j$, which can occur with at most negligible probability according to Lemma 3.

Next, we prove the ID-concealed session-key (ICSK) security. Let the test-session sid_T, held at the uncorrupted user $U_t = U_{t_\sigma}$ for $\sigma \leftarrow \{0,1\}$, is completed and unexposed. We first prove the *impersonation* security. Specifically, suppose U_t is an initiator (client) user and $peer_{sid_T} = U_k \in \mathsf{HONEST}$ where U_k is a responder user, but sid_T has no matching session. Note that, for the test-session sid_T run at Initiator to be unexposed, the following must hold simultaneously: (1) The static secret-key of the peer user U_k is unexposed. (2) The local state of ST_{sid_T} is unexposed (*but the local state of* $\mathsf{ST}_{sid'_T}$ *of the matching session can be exposed*). (3) Neither U_{t_0} nor U_{t_1} is corrupted. (4) Neither the session-key of sid_T nor that of sid'_T (in case the matching session exists) is exposed. This implies that a PPT adversary can impersonate the honest responder user U_k.

Denote by $(X, \overline{Y'})$ the session label of sid_T, and by (H'_B, C'_B) the AEAD ciphertext sent by $\mathcal{A}$ in the second round of the test-session. As we assume sid_T has been

successfully completed, it means that (H'_B, C'_B) was decrypted to (pid_k, Y') such that $\overline{Y}' = pk_k Y'^{h(X,Y',pid_k)}$, where Y' may be generated by $\mathcal{A}$ itself. We consider two cases.

The first case, referred to as Case-1, is that $\overline{Y}'$ was sent by U_k in a session of label $(X', \overline{Y}')$, where $X' \neq X$ as we assume no matching session exists for sid_T. In this case, U_k computed $\overline{Y}' = pk_k Y^{h(X',Y,pid_k)}$ where $Y = g^y$ for $y \leftarrow Z_q^*$ taken by U_k itself. This means that $\overline{Y}' = pk_k Y'^{h(X,Y',pk_k)} = pk_k Y^{h(X',Y,pid_k)}$ where $X \neq X'$, which occurs with at most negligible probability according to Lemma 3.

The second case, referred to as Case-2, is $\overline{Y}'$ was never sent by U_k (but may be sent by another user in another session unmatched to sid_T). We have the following lemma, which then establishes the *impersonation* security.

Lemma 4. *Case-2 occurs with at most negligible probability, under the* AEAD *security and the GDH assumption in the random oracle model.*

Proof (of Lemma 4). Assuming there exists a PPT adversary $\mathcal{A}$ such that Case-2 occurs with non-negligible probability p, we construct another algorithm $\mathcal{S}$ who can break the GDH assumption also with non-negligible probability under the AEAD security in the RO model. $\mathcal{S}$ takes input (B, X), where $B = g^b$ and $X = g^x$ for $b, x \leftarrow Z_q^*$ that are unknown to $\mathcal{S}$, and its goal is to compute $CDH(B, X)$ with a DDH oracle. We assume $\mathcal{A}$ runs at most s sessions at either Initiator or Responder, where s is polynomial in $|q|$. $\mathcal{S}$ randomly guesses the victim responder user U_k by taking $k \leftarrow \{1, \cdots, n\}$, hoping U_k is the victim responder user. With probability $\frac{1}{n}$, the guess of $\mathcal{S}$ is correct. For presentation simplicity, we view the victim U_k as user of public identity information pid_B. $\mathcal{S}$ sets the public-key of U_k to be the element B given in its input, i.e., $pk_k = B$, and registers $(id_k, B) = (id_B, B)$ to get the certificate $cert_k = cert_B$ via CA. $\mathcal{S}$ sets the public-key and secret-key for any user U_j, $1 \leq j \neq k \leq n$, by itself, and will act on its behalf. $\mathcal{S}$ embeds a random bit $\sigma \leftarrow \{0, 1\}$, and runs $\mathcal{A}$ as a subroutine, and answers its oracle queries as follows.

When $\mathcal{A}$ starts a session with the responder U_k via the oracle query "$(\mathsf{Start}, U_k, X')$", where $X' \in G$, $\mathcal{S}$ works as follows: It generates $Y = g^y$, $y \leftarrow Z_q^*$, computes $\overline{Y} = BY^{h(X',Y,pid_B)}$, takes $(K_1, K_2) \leftarrow \{0,1\}^\kappa \times \{0,1\}^\kappa$, and sends $\{H_B, \overline{Y}, C_B = \mathsf{Enc}_{K_1}(H_B, pid_B \| Y)\}$ to $\mathcal{A}$. $\mathcal{S}$ also stores $\{X' || \overline{Y}, (K_1, K_2)\}$ into a list denoted $\mathcal{L}_{DDH}$.[18] Note that S cannot compute the pre-shared secrecy $PS = CDH(X', \overline{Y})$ and consequently $KDF(PS, X' || \overline{Y})$, as it does not know the secret-key b. In order to keep the consistency of the random oracle KDF, from now on whenever the adversary $\mathcal{A}$ makes an oracle query of the form $KDF(PS', X' || \overline{Y})$, based on the stored list $\mathcal{L}_{DDH}$ the simulator S checks whether $PS'/X'^{yh(X',\overline{Y},pid_B)} =$

[18] In the actual analysis, if $\{X' || \overline{Y}, (K_1, K_2)\}$ has already been defined previously in $\mathcal{L}_{DDH}$, $\mathcal{S}$ will abort, which occurs with at most negligible probability.

$CDH(B, X')$ with the aid of its DDH oracle $DDH(B, \cdot, \cdot)$; if yes, it returns the pre-set value (K_1, K_2).

Upon receiving "$(\mathsf{Test}, U_{t_0}, U_{t_1})$" for uncorrupted $U_{t_0}, U_{t_1} \in \mathsf{HONEST}$, where $1 \leq t_0 \neq t_1 \leq n$, it sets $U_t = U_{t_\sigma}$, $\mathcal{S}$ just sends X, the element given in its input, to $\mathcal{A}$ as the first-round message of the test-session.

As $\mathcal{S}$ knows the secret-keys of all the other users except U_k, and Case-2 assumes that secret-key of U_k is unexposed and no test-session exists for sid_T, $\mathcal{S}$ can perfectly answer all the other queries made by $\mathcal{A}$ regarding StaKey, $\mathsf{Corrupt}$, $\mathsf{Register}$ and Peer. It is easy to check that the view of $\mathcal{A}$ under the run of $\mathcal{S}$ is identical to that in its real attack, where KDF is assumed to be an RO. This means, with non-negligible probability, Case-2 occurs in the simulation of $\mathcal{S}$.

Denote by $(X, \overline{Y}')$ the session label of the completed test-session sid_T, and by (H'_B, C'_B) the AEAD ciphertext sent by $\mathcal{A}$ in the second round of sid_T. Case-2 means that $\overline{Y}'$ was not generated by user pid_B, but the decryption of (H'_B, C'_B) gives (pid_B, Y') such that $\overline{Y}' = BY'^e$ for $e = h(X, Y', pid_B)$, where $Y' = g^{y'}$ may be generated by $\mathcal{A}$ itself. It also implies that $\mathcal{A}$ has made the oracle query $h(X, Y', B)$ to get e, as, otherwise, $\overline{Y}' = BY'^e$ holds (and consequently the test-session can be successfully finished) only with negligible probability. Denote by $(K'_1, K'_2) = KDF(CDH(X, \overline{Y}'), X||\overline{Y}')$. As we assume sid_T has no matching session and $\overline{Y}'$ was not sent by pid_B in Case-2, with overwhelming probability the AEAD ciphertext $\mathsf{Enc}_{K'_1}(H'_B, pid_B||Y')$ was not sent in any existing session other than the test-session. By the AEAD security, the adversary $\mathcal{A}$ must have made the oracle query $KDF(CDH(X, \overline{Y}'), X||\overline{Y}')$, from which $\mathcal{S}$ gets $CDH(X, \overline{Y}') = X^{b+y'e}$.

$\mathcal{S}$ rewinds $\mathcal{A}$ to the point of RO query $h(X, Y', pid_B)$, and redefines $h(X, Y', pid_B) = e' \leftarrow \{0,1\}^l \bigcap Z_q^*$, and runs $\mathcal{A}$ from this rewinding point, where all RO queries to h after the rewinding point are answered randomly and independently. According to the general forking lemma [4], Case-2 occurs in the rewound run also with non-negligible probability, from which $\mathcal{S}$ will get $CDH(X, \overline{Y}') = X^{b+y'e'}$. Note that, with overwhelming probability, $e' \neq e$. By computing $X^{y'} = (X^{b+y'e}/X^{b+y'e'})^{e-e'}$ and $X^{b+y'e}/X^{y'e} = X^b = CDH(X, B)$, $\mathcal{S}$ breaks the GDH assumption also with non-negligible probability. □

Finally, we prove the ID-SK indistinguishability. For UCAKE protocol, we only need to consider the ID-indistinguishability when the completed unexposed test-session sid_T is run at $\mathsf{Responder}$. Note that, for the completed test-session sid_T run at $\mathsf{Responder}$ to be unexposed, the following conditions must hold simultaneously: (1) The matching session sid'_T exists at an uncorrupted honest client user U_j. (2) The adversary didn't make the query $\mathsf{Peer}(sid'_T)$. (3) Neither U_{t_0} nor U_{t_1} is corrupted, *but the static secret-keys of both* U_{t_0} *and* U_{t_1} *can be exposed.* (4) Neither

ST_{sid_T} nor $ST_{sid'_T}$ is exposed. （5）Neither the session-key of sid_T nor that of sid'_T is exposed.

In this case, the first observation is that $\overline{Y}$ perfectly hides the responder's identity in the random oracle model. Specifically, for any $\sigma \in \{0,1\}$, any $X' \in G$, and any pair of (pid_{t_0}, pid_{t_1}), where $1 \leq t_0, t_1 \leq n$ and sk_{t_0} (resp., sk_{t_1}) is the static secret-key of pid_{t_0} (resp., pid_{t_1}), it is easy to check that the following distributions are identical in the random oracle model assuming h is an RO: $\{(pid_{t_0}, pid_{t_1}), (sk_{t_0}, sk_{t_1}), X', \overline{Y}_\sigma = pk_{t_\sigma} Y^{h(X', Y, pid_{t_\sigma})}\}$ and $\{(pid_{t_0}, pid_{t_1}), (sk_{t_0}, sk_{t_1}), X', \widetilde{Y}\}$, where $\widetilde{Y} \leftarrow G \setminus 1_G$ and $Y = g^y$ for $y \leftarrow Z_q^*$ that is assumed to be unexposed. It can also be easily seen that, if y is taken from a well-spread distribution over Z_q^* with min-entropy greater than $\omega(\log |q|)$, $\overline{Y}$ *computationally* hides the responder's identity under the FGDL assumption in the random oracle model.

Then, by the AEAD security, to break the ID-SK indistinguishability, in the random oracle model assuming KDF is an RO, $\mathcal{A}$ has to query KDF in order to get the session-key of the test-session or its matching session. We further examine two cases.

The first case is that the completed unexposed test-session sid_T is run at Responder. In this case, as the matching session sid'_T must exist and neither ST_{sid_T} nor $ST_{sid'_T}$ is exposed, we show that the ability to break ID-SK indistinguishability implies the ability to break the CDH assumption. Specifically, the simulator $\mathcal{S}$ takes input (X, Y), where $X, Y \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(X, Y)$ with the aid of a DDH-oracle. Toward this goal, it sets the public-keys and secret-keys for all the honest users, and runs $\mathcal{A}$ as a subroutine. $\mathcal{S}$ randomly guesses, with successful probability at least $\frac{1}{s}$, the session sid'_T matching to the test-session sid_T, by randomly taking $i \leftarrow \{1, \cdots, s\}$, where $\mathcal{A}$ is assumed to run at most s sessions at either Initiator or Responder. For the i-th session run at Ininiator (i.e., the guessed matching session sid'_T), $\mathcal{S}$ sends X (i.e., the value given in its input) in the first round. When $\mathcal{A}$ makes the query "$(\mathsf{Test}, U_{t_0}, U_{t_1}, X)$" for $U_{t_0}, U_{t_1} \in \mathsf{HONEST}$ regarding the test-session sid_T, $\mathcal{S}$ sets $U_t = U_{t_\sigma}$ where $\sigma \leftarrow \{0,1\}$ is the random bit embedded in $\mathcal{S}$, and sends $\overline{Y} = BY^e$ (and the accompanying AEAD ciphertext) in the second round of sid_T, where $e = h(X, Y, pid_B)$ and Y is the value given in the input of $\mathcal{S}$. For presentation simplicity, denote by pid_B (with public-key $B = g^b$ where $b \leftarrow Z_q^*$ is generated by $\mathcal{S}$ itself) the responder user U_t who runs the test-session sid_T. To break the ID-SK indistinguishability, the above reasoning has established that $\mathcal{A}$ has to make the RO query $KDF(CDH(X, \overline{Y}), X||\overline{Y})$, from which $\mathcal{S}$ computes $CDH(X, Y) = (CDH(X, \overline{Y})/X^b)^{e^{-1}}$.

The second case is that the completed unexposed test-session sid_T is run at Initiator. Recall that, for the test-session sid_T run at Initiator to be unexposed, the following must hold simultaneously: (1) The static secret-key of the peer user U_k is unexposed. (2) The local state of ST_{sid_T} is unexposed (*but the local state of* $\mathsf{ST}_{sid'_T}$

of the matching session can be exposed). (3) Neither U_{t_0} nor U_{t_1} is corrupted. (4) Neither the session-key of sid_T nor that of sid'_T is exposed. For presentation simplicity, we assume the peer user U_k is pid_B. Also note that the impersonation security has already established that the matching session sid' must exist at the responder user pid_B. In this case, we show that the ability to break ID-SK indistinguishability implies the ability to break the GDH assumption. Specifically, the simulator $\mathcal{S}$ takes input (X, B), where $X, B \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(X, B)$ with the aid of a DDH-oracle. Toward this goal, $\mathcal{S}$ randomly guess the peer user pid_B with successful probability $\frac{1}{n}$, whose public-key is just set to be B, and sets the public-keys and secret-keys for all the other honest users in the system. $\mathcal{S}$ runs $\mathcal{A}$ as a subroutine, and works as follows.

- As in the proof of Lemma 4, $\mathcal{S}$ perfectly simulates the user pid_B with the aid of the DDH-oracle, particularly ensuring the consistency in answering the queries by $\mathcal{A}$ to the random oracle KDF.
- $\mathcal{S}$ sends X (i.e., the value given in its input) in the first round of sid_T, upon receiving the query "$(\mathsf{Test}, U_{t_0}, U_{t_1})$" from $\mathcal{A}$.
- For any session (including the matching session sid'_T) run by pid_B at $\mathsf{Responder}$, $\mathcal{S}$ sends $\overline{Y} = BY^e$ in the second round, where $e = h(X', Y, pid_B)$ and $Y = g^y$ for $y \leftarrow Z_q^*$ that is generated by $\mathcal{S}$ itself but can be exposed to $\mathcal{A}$. Note that in the matching session sid'_T, it holds that $X' = X$.

Again, in order to break the ID-SK indistinguishability in this case, the above reasoning has established that $\mathcal{A}$ has to make the RO query $KDF(CDH(X, \overline{Y}), X||\overline{Y})$, from which $\mathcal{S}$ computes $CDH(X, B) = CDH(X, \overline{Y})/X^{ye}$. This finishes the proof of Theorem 5. □

Corollary 2. *Let $\mathcal{Y}$ be a well-spread distribution over Z_q^* with min-entropy $\lambda_{\mathcal{Y}} > \omega(\log |q|)$. The protocol described in Fig. 5, when y is taken according to $\mathcal{Y}$, is UCAKE-secure, under the AEAD security, the FGDH (actually HGDH) and FGDL assumptions (as defined in Section 3), in the random oracle model.*

Corollary 3 indicates much flexible implementations of the UCAKE protocol, according to priorities and tradeoffs among security and efficiency. In practice, at one's own discretion (according to task criticality and application scenarios), the responder can take $y \leftarrow \{0,1\}^{\lceil |q|/4 \rceil} \cap Z_q^*$ (referred to as *light*-UCAKE), or $y \leftarrow \{0,1\}^{\lceil |q|/2 \rceil} \cap Z_q^*$ (referred to as *medium*-UCAKE), or just $y \leftarrow Z_q^*$ (referred to as *full*-UCAKE). For example, if $|q| = 512$, we suggest $|y| = 128 = |q|/4$, i.e., *light*-UCAKE, could suffice for many application scenarios. In this case, the responder only performs about 1.75 exponentiations (compared to 2.5 exponentiations in *full*-higncryption). For most application scenarios, we may recommend to use *medium* UCAKE (M-UCAKE, for short). Note that, for M-UCAKE, computing $CDH(X, \overline{Y} = BY^{h(X,Y,pid_B)})$ is not reduced to solving half a DL problem

w.r.t. Y where $y \leftarrow \{0,1\}^{\lceil |q|/2 \rceil} \cap Z_q^*$, even if the static secret-key b is exposed. Actually, given $Y^{h(X,Y,pid_B)}$, it seems that no efficient way to even compute the DH-component Y (letting alone the DH-exponent y), other than exclusively searching over $y \leftarrow \{0,1\}^{\lceil |q|/2 \rceil} \cap Z_q^*$.

9.3 Comparison and Discussion

A brief comparison among M-UCAKE, UCAKE, QUIC and DSA-TLS1.3 is given in Table 3. Here, QUIC refers to the basic protocol structure presented in Fig. 11 (page 59). DSA-TLS1.3 refers to the SIGMA-based basic authentication mechanism of TLS1.3 described in Fig. 12, when the underlying server signature is implemented with DSA.

For efficiency comparison, we only count the number of modular exponentiations (denoted as "exp." in the table). By "y-security", we mean that the session-key remains secure even if server's DH-exponent y is exposed. By "forward-ID", we mean that server's identity privacy holds even when server's static secret-key is exposed. By "sig-vulnerability", we mean that the exposure of random nonce generated during signature computation will leak server's static secret-key.

		M-UCAKE	UCAKE	QUIC	DSA-TLS 1.3
efficiency	server	2exp.	2.5 exp.	3 exp.	3 exp.
	client	2.5 exp.	2.5 exp.	3 exp.	3.5 exp.
y-security		$\surd$	$\surd$	X	X
forward-ID		$\surd$	$\surd$	X	$\surd$
sig-vulnerability		$\perp$	$\perp$	$\perp$	$\surd$

Table 3: Comparison among M-UCAKE, UCAKE, QUIC and DSA-TLS1.3

9.4 Full TLS Handshake: The Second Solution

In this section, based upon UCAKE, we develop more efficient protocol construction for full TLS handshake (with optionally 0.5-RTT authentication). The basic protocol structure is presented in Fig. 6.

In the protocol description, "$\{\ \}_K$" denotes AEAD encryption using key K, "$\{\ \}_K^*$" (resp., "$\{\ \}_K^+$") means it is optionally generated and sent only for 0-RTT (resp., 0.5-RTT) mode. τ_A and τ_B denote some auxiliary information (like random nonces, protocol version, extensions, system parameters, timing stamps, etc) sent by pid_A and pid_B respectively, though they may not be necessary for provable security. SFIN and CFIN are the "finish" messages sent by server and client respectively, as is done in TLS1.3 or QUIC. We note that SFIN and CFIN (particularly, SFIN) may not be necessary for the provable security of UCAKE-based

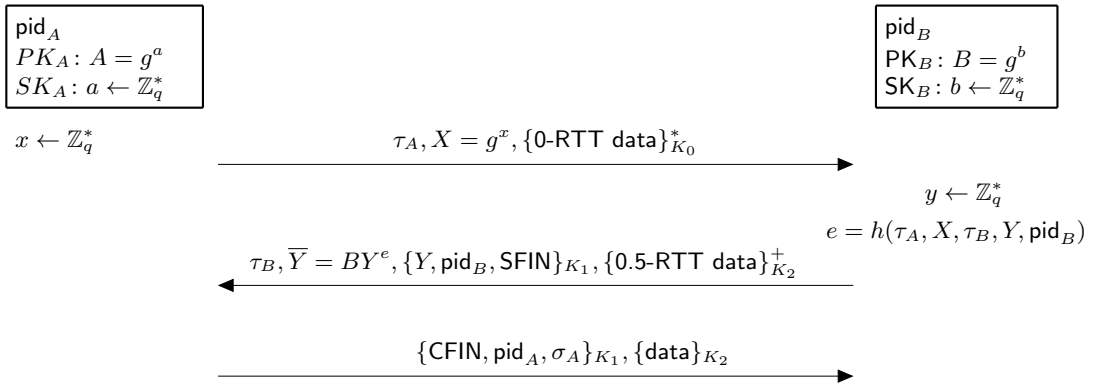

Fig. 6: TLS handshake based on UCAKE

TLS. σ_A is the signature of the client on the hash of (plaintext) transcript up to $\{\mathsf{CFIN}, pid_A\}$. Here, (pid_A, σ_A) is sent only when server indicates client authentication in the second round. The key K_0 is derived from $CDH(X, B)$ and public transcript including (pid_A, pid_B, X) and even pre-shared key if it is run in the resumption model. The keys K_1 and K_2 are derived from $PS = CDH(X, \overline{Y})$ (and public transcript), in a way that K_1 and K_2 are computationally independent as is done in TLS1.3 or QUIC.[19]

10 Protocol Construction and Security Analysis of CAKE with Mutual Authentication

In this section, we present the protocol constructions and security analysis for identity-concealed authenticated key-exchange with mutual authentications, referred to as CAKE for presentation simplicity.

The basic protocol structure of CAKE is presented in Fig. 7, where the system parameters are the same as in the construction of UCAKE. For a session run, the local state ST_{sid} is specified to be the DH-exponent y (if sid is run at Responder) or x (if sid is run at Initiator), which can be offline computed and stored. We assume that each user always performs, explicitly or implicitly, subgroup membership test for the incoming DH-exponent $\overline{X}$ or $\overline{Y}$.

Again, at one's own discretion (according to task criticality and application scenarios), user pid_A (resp., pid_B) can select $x \leftarrow \{0,1\}^{l_A}$ (resp., $y \leftarrow \{0,1\}^{l_B}$), where $\omega(\log|q|) < l_A, l_B \leq |q|$, and l_A and l_B may not necessarily be equal. For example, user pid_A can set, according to task criticality and application scenarios, $l_A = \lceil |q|/4 \rceil$, or $l_A = \lceil |q|/2 \rceil$, or $l_A = \lceil 3|q|/4 \rceil$, or just $l_A = |q|$. In practice, it may

[19] If 0.5-RTT authentication is not run, the session-key K_2 can be derived from PS and (the hash of) the transcript up to $\{\mathsf{data}\}_{K_2}$.

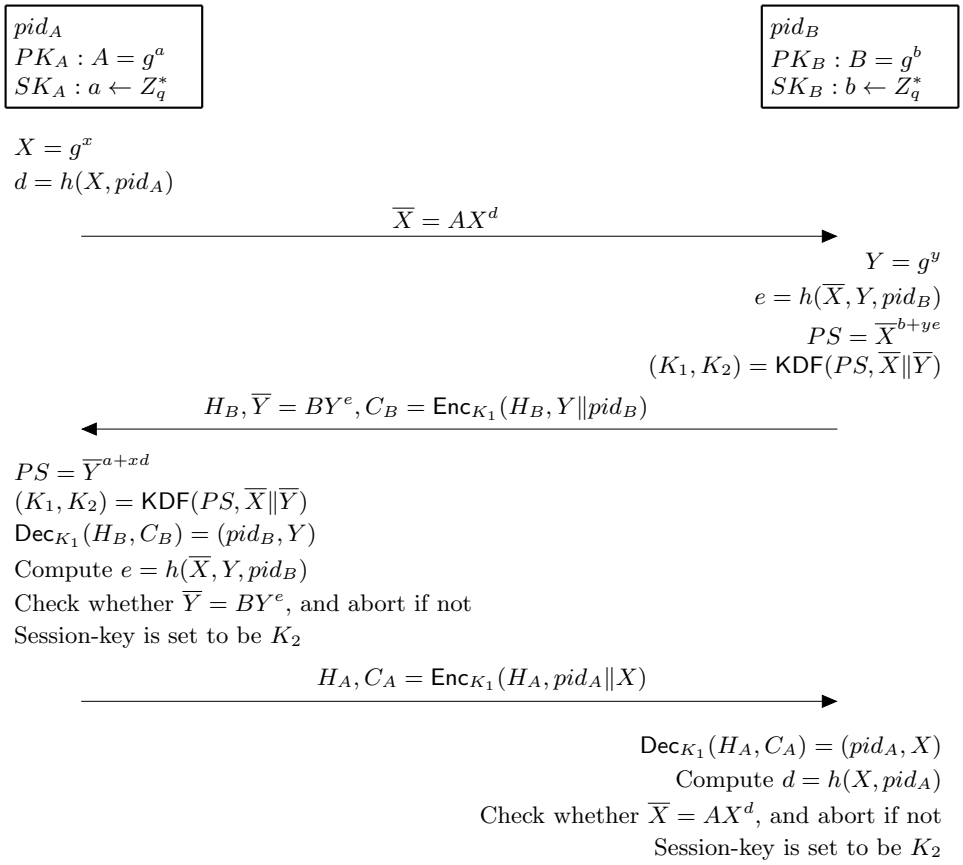

Fig. 7: Basic Protocol Structure of CAKE

be recommended to set $l_A = l_B = \lceil |q|/2 \rceil + 1$. The session-key K_2 could also be set to be K_1 (and in this case the provable security will rely upon the ACCE security model), or $K_2 = KDF(PS, X||pid_A||Y||pid_B)$. A generalized protocol structure of CAKE is presented in Fig. 16, Appendix B (page 61).

10.1 Security Analysis of CAKE

Theorem 6. *The protocol described in Fig. 7 is* strongly *CAKE-secure according to Definition 1, under AEAD security and the GDH assumption in the Random oracle model.*

Proof. For each session run of the CAKE protocol (at either Initiator or Responder), define the session label to be "$\overline{X}||\overline{Y}$". Two sessions (whether they are complete or not) are matching if they have the same session label. Note that, for honestly generated $\overline{X}$ and $\overline{Y}$, with overwhelming probability $\overline{X} \neq \overline{Y}$, which

holds even if x and y are taken from some well-spread distributions over Z_q^* with min-entropy greater than $\omega(\log |q|)$.

We first prove the label-security. Firstly, it is trivial to check that the probability that more than two sessions share the same session label is negligible. Now, consider any pair of matching sessions: session sid held at user U_i and session sid' held at user U_j. Label matching (with inequality between $\overline{X}$ and $\overline{Y}$) implies that, with overwhelming probability, U_i and U_j cannot play the same session role. As $PS = CDH(\overline{X}, \overline{Y})$ and session-key is derived from $KDF(PS, X||\overline{Y})$, we conclude that the two matching sessions must have the same session-key.

Without loss of generality, assume that U_i (holding session sid) is initiator and U_j (holding the matching session sid') is responder. Next, we prove that the probability of $peer_{sid} \neq \perp \wedge peer_{sid} \neq U_j$, or $peer_{sid'} \neq \perp \wedge peer_{sid'} \neq U_i$, is negligible. Note that, for CAKE protocol, it may be the case that $peer_{sid} = U_j$ but $peer_{sid'} = \perp$. Specifically, consider that the third-round message sent by the initiator U_i in session sid is dropped by adversary, which causes $peer_{sid} = U_j$ but $peer_{sid'} = \perp$. We note that such a cutting-last-message attack is unavoidable. Let $\overline{X} = pk_i X^{h(X,pid_i)}$ (sent by U_i in session sid) and $\overline{Y} = pk_j Y^{h(\overline{X},Y,pid_j)}$ (sent by U_j in the matching session sid'), and suppose that $peer_{sid} \neq \perp \wedge peer_{sid} \neq U_j$, or $peer_{sid'} \neq \perp \wedge peer_{sid'} \neq U_i$. This implies that there exists a PPT adversary who can successfully open $\overline{Y} = pk_j Y^{h(\overline{X},Y,pid_j)}$ into $(\overline{X}, Y', pid_k)$ for $pid_k \neq pid_j$ in session sid such that $\overline{Y} = pk_j Y^{h(\overline{X},Y,pid_j)} = pk_k Y'^{h(\overline{X},Y',pid_k)}$, or can open $\overline{X} = pk_i X^{h(X,pid_i)}$ into (X', pid_δ) for $pid_\delta \neq pid_i$ in session sid' such that $\overline{X} = pk_i X^{h(X,pid_i)} = pk_\delta X'^{h(X',pid_\delta)}$. However, according to Lemma 3, either case can occur with at most negligible probability.

Let sid_T be the *completed unexposed* test-session held at the user $U_t = U_{t_\sigma}$ with session-label $\overline{X}||\overline{Y}$ and $peer_{sid_T} = U_k \in \mathsf{HONEST}$, $1 \leq k \leq n$, where U_t may be identical to U_k. Denote by sid'_T its matching session (in case it exists), which may be still on-going. As the test-session sid_T is completed and unexposed, it means that:

- U_{t_0} and U_{t_1} are not corrupted (but the static secret keys of them could be exposed), and the static secret-key of $peer_{sid_T} = U_k$ is unexposed.
- The local state ST_{sid_T} is unexposed if $U_t \neq U_k$ (otherwise, ST_{sid_T} is allowed to be exposed), but $ST_{sid'_T}$ may be exposed.
- The session-key of either sid_T or sid'_T is unexposed;
- $peer_{sid'_T}$ is unexposed.

The ID-concealed session-key security is reduced to the AEAD security and the GDH assumption in the random oracle model. Before proceeding the security analysis, we first clarify the use of the DDH-oracle in ensuring the consistency of the KDF random oracle. Specifically, we consider a simulator $\mathcal{S}$ who wants to

simulate, with the aid of a DDH-oracle, the peer user $peer_{sid_T}$ of the test-session, as well as the user U_t in the test-session sid_T.

- If $peer_{sid_T}$, which is denoted pid_B of public-key $B = g^b$ in this case for presentation simplicity, runs (holds) a session sid in which it plays the responder role, $\mathcal{S}$ gets access to the DDH-oracle $DDH(B, \cdot, \cdot)$, and acts as follows. Note that, when $peer_{sid_T} = U_t$, sid can just be the test-session sid_T. After receiving $\overline{X}$ in the first round, $\mathcal{S}$ generates $Y = g^y$ and $\overline{Y} = BY^e$ in the second round where $e = h(\overline{X}, Y, pid_B)$; $\mathcal{S}$ randomly generates $(K_1, K_2) \leftarrow \{0,1\}^\kappa \times \{0,1\}^\kappa$ by itself, records $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ in a list denoted $\mathcal{L}_{DDH}$,[20] uses K_1 to get the AEAD ciphertext (H_B, C_B), and finally sends $\{\overline{Y}, H_B, C_B\}$ in the second round of the session. From this point on, for each RO-query of the form $KDF(PS, \overline{X}||\overline{Y})$ made by the CMIM adversary $\mathcal{A}$, where PS is supposed to be $CDH(\overline{X}, \overline{Y}) = \overline{X}^{b+ye}$, $\mathcal{S}$ computes $Z = PS/\overline{X}^{ye}$, and queries its DDH-oracle with $(B, \overline{X}, Z)$: if the DDH-oracle outputs "yes" $\mathcal{S}$ returns the already stored (K_1, K_2) to the adversary; otherwise, random answer is returned.
- If $peer_{sid_T}$, which is denoted pid_A of public-key denoted $A = g^a$ in this case for presentation simplicity, runs/holds a session sid in which it plays the initiator role, $\mathcal{S}$ gets access to the DDH-oracle $DDH(A, \cdot, \cdot)$ and acts as follows. Note that, when $peer_{sid_T} = U_t$, sid can just be the test-session sid_T. $\mathcal{S}$ generates $X = g^x$ and sends $\overline{X} = AX^d$ in the first round where $d = h(X, pid_A)$. After receiving the second-round message denoted $\{\overline{Y}, H_B, C_B\}$, $\mathcal{S}$ first checks whether $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ has already been stored in $\mathcal{L}_{DDH}$: if "yes", $\mathcal{S}$ just uses K_1 to decrypt C_B; if "not", $\mathcal{S}$ randomly generates $(K_1, K_2) \leftarrow \{0,1\}^\kappa \times \{0,1\}^\kappa$ by itself and records $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ into the list $\mathcal{L}_{DDH}$. From this point on, for each RO-query of the form $KDF(PS, \overline{X}||\overline{Y})$ made by the CMIM adversary $\mathcal{A}$, where PS is supposed to be $CDH(\overline{X}, \overline{Y}) = \overline{Y}^{a+xd}$, $\mathcal{S}$ computes $Z = PS/\overline{Y}^{xd}$, and queries its DDH-oracle with $(A, \overline{Y}, Z)$: if the DDH-oracle outputs "yes" $\mathcal{S}$ returns the already stored (K_1, K_2) to the adversary; otherwise, random answer is returned.
- Simulation of the test-session sid_T. Note that the above analysis has already dealt with the test-session simulation when $peer_{sid_T} = U_t$. Here, we only consider the case of $peer_{sid_T} \neq U_t$.
  If sid_T is run at Responder, denote by $U_t = pid_B$ the holder of the test-session for presentation simplicity. For this case, the public-key $B = g^b$ and secret-key $b \leftarrow Z_q^*$ of pid_B are generated by the simulator $\mathcal{S}$ itself, as the secret-key b may be exposed to adversary. $\mathcal{S}$ takes $Y \leftarrow G \setminus 1_G$ as input, and has access to the DDH-oracle $DDH(Y, \cdot, \cdot)$. After receiving $\overline{X}$ in the first round of sid_T, $\mathcal{S}$ generates $\overline{Y} = BY^e$ where $e = h(\overline{X}, Y, pid_B)$ and Y is the value

[20] In the actual analysis, if $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ has already been defined previously, $\mathcal{S}$ will abort, which occurs with at most negligible probability.

given in $\mathcal{S}$'s input, and checks whether $\{\overline{X}||\overline{Y}, (K_1, k_2)\}$ has been recorded in $\mathcal{L}_{DDH}$: if "yes" $\mathcal{S}$ will abort, which occurs with at most negligible probability in the RO model; if "not", $\mathcal{S}$ randomly generates (K_1, K_2) by itself and records $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ into the list $\mathcal{L}_{DDH}$. From this point on, for each RO-query of the form $KDF(PS, \overline{X}||\overline{Y})$ made by the CMIM adversary $\mathcal{A}$, where PS is supposed to be $CDH(\overline{X}, \overline{Y}) = \overline{X}^{b+ye}$, $\mathcal{S}$ computes $Z = (PS/\overline{X}^b)^{e^{-1}}$, and queries its DDH-oracle with $(Y, \overline{X}, Z)$: if the DDH-oracle outputs "yes", $\mathcal{S}$ returns the already stored (K_1, K_2) to the adversary; otherwise, random answer is returned.

If the test-session sid_T is run at Initiator, denote by $U_t = pid_A$ the test-session holder for presentation simplicity. For this case, the public-key $A = g^a$ and secret-key a of pid_A are generated by the simulator $\mathcal{S}$ itself, as the secret-key a can be exposed to adversary. $\mathcal{S}$ takes $X \leftarrow G \setminus 1_G$ as input, and gets access to the DDH-oracle $DDH(X, \cdot, \cdot)$. In the first round of sid_T, $\mathcal{S}$ sends $\overline{X} = AX^d$, where $d = h(X, pid_A)$ and X is the value given in $\mathcal{S}$'s input. After receiving $\{\overline{Y}, (H_B, C_B)\}$ in the second round of the test-session, $\mathcal{S}$ checks whether $\{\overline{X}||\overline{Y}, (K_1, k_2)\}$ has been recorded in $\mathcal{L}_{DDH}$: if "yes" $\mathcal{S}$ just uses K_1 to decrypt C_B, and proceeds further; if "not", $\mathcal{S}$ randomly generates $(K_1, K_2) \leftarrow \{0,1\}^\kappa \times \{0,1\}^\kappa$ by itself and records $\{\overline{X}||\overline{Y}, (K_1, K_2)\}$ into the list $\mathcal{L}_{DDH}$. From this point on, for each RO-query of the form $KDF(PS, \overline{X}||\overline{Y})$ made by $\mathcal{A}$, where PS is supposed to be $CDH(\overline{X}, \overline{Y}) = \overline{Y}^{a+xd}$, $\mathcal{S}$ computes $Z = (PS/\overline{Y}^a)^{d^{-1}}$, and queries its DDH-oracle with $(X, \overline{Y}, Z)$: if the DDH-oracle outputs "yes" $\mathcal{S}$ returns the already stored (K_1, K_2) to the adversary; otherwise, random answer is returned.

It is easy to check that,in the random oracle model where KDF is assumed to be an RO, with overwhelming probability the simulation of $peer_{sid_T}$ and the test-session by $\mathcal{S}$ is perfect.

Next, we prove the impersonation security. Suppose an efficient adversary $\mathcal{A}$ could successfully impersonate the honest user U_k in the completed and unexposed test-session sid_T, while no matching session exists. We distinguish two cases according to whether sid_T is run at Initiator or Responder, and present the outline of proof based on the proof of Theorem 5.

The first case, denoted Case-1, is that sid_T is run at Initiator. In this case, for presentation simplicity, we denote by $U_t = pid_A$ the test-session holder and by $U_k = pid_B$ the peer user $peer_{sid_T}$. The proof in this case is similar to that in the proof of Theorem 5 for UCAKE. Firstly, by Lemma 3, we have that with overwhelming probability $\overline{Y}$ was never generated and sent by pid_B in any existing session. Then, the impersonation security is reduced to the AEAD security and the GDH assumption in the RO model. Specifically, the GDH-solver $\mathcal{S}$ takes (B, X) as input, where $B, X \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(B, X)$ with

the aid of DDH-oracle. Towards this goal, it randomly guesses the the peer user $peer_{sid_T} = pid_B$ with successful probability $\frac{1}{n}$, generates the public-keys and secret-keys for all the honest users in the system except pid_B whose public-key is set to be B (i.e., the value given in $\mathcal{S}$'s input). Similar to the proof of Theorem 5 and as clarified above regarding the simulation of $peer_{sid_T}$ and the test-session, $\mathcal{S}$ can well simulate the actions of pid_B. In the test-session sid_T, $\mathcal{S}$ sets and sends $\overline{X} = AX^d$ for $d = h(X, pid_A)$ in the first round, where X is the value given in $\mathcal{S}$'s input. Denote by $(\overline{Y}, H_B, C_B)$ the messages sent by the CMIM adversary $\mathcal{A}$ in the second round of the test-session sid_T. As the test-session is completed and unexposed but has no matching session, by the AEAD security in the RO model, $\mathcal{A}$ has to make the RO query $KDF(PS, \overline{X}||\overline{Y})$, where it holds that $\overline{Y} = BY^e$ for some Y (encrypted in the AEAD ciphertext C_B) and $e = h(\overline{X}, Y, pid_B)$, from which $\mathcal{S}$ can get $PS = CDH(\overline{X}, \overline{Y}) = \overline{Y}^{a+xd}$.

For now, we assume $pid_A \neq pid_B$, and thus $\mathcal{S}$ knows the secret-key a of pid_A. Denoting by $B = g^b$, $X = g^x$ and $Y = g^y$, $\mathcal{S}$ can then compute $CDH(X, \overline{Y}) = (PS/\overline{Y}^a)^{d^{-1}} = \overline{Y}^x = (BY^e)^x = B^x Y^{ex}$. Note that, with overwhelming probability, the RO-query $d = h(X, pid_A)$, made by $\mathcal{S}$ itself, is prior to the RO-query $e = h(\overline{X}, Y, pid_B)$. Then, by rewinding $\mathcal{A}$ and re-programming the random oracle $h(\overline{X}, Y, pid_B) = e'$, with non-negligible probability $\mathcal{A}$ will send $\overline{Y}' = BY^{e'}$ in the second round of the test-session sid_T and successfully finish the test-session in the rewound run according to the general forking lemma [4]. Again, by the AEAD security in the RO model, $\mathcal{S}$ will obtain $PS' = CDH(\overline{X}, \overline{Y}') = \overline{Y}'^{a+xd}$, from which it can get $CDH(X, \overline{Y}') = (PS'/\overline{Y}'^a)^{d^{-1}} = \overline{Y}'^x = B^x Y^{e'x}$. Then, $\mathcal{S}$ computes $CDH(X, Y) = (CDH(X, \overline{Y})/CDH(X, \overline{Y}'))^{(e-e')^{-1}}$. Finally, $\mathcal{S}$ computes $CDH(B, X) = CDH(X, \overline{Y})/CDH(X, Y)^e$, which violates the GDH assumption.

Note that the above proof of Case-1 assumes $pid_A \neq pid_B$. If $pid_A = pid_B$, the input to $\mathcal{S}$ will only be $B = g^b$, where $b \leftarrow Z_q^*$ that is unknown to $\mathcal{S}$, and its goal is to compute $CDH(B, B) = g^{b^2}$, which is as hard as solving the CDH problem [32]. In the test-session sid_T, $\mathcal{S}$ generates $X = g^x$ by itself (and thus knowing $x \leftarrow Z_q^*$), and sends $\overline{X} = AX^d = BX^d$ in the first round where $d = h(X, pid_A) = h(X, pid_B)$. Similarly, by AEAD security in the RO model, $\mathcal{S}$ can get $PS = CDH(\overline{X}, \overline{Y}) = \overline{Y}^{b+xd} = (BY^e)^{b+xd}$ and compute $CDH(B, \overline{Y}) = PS/\overline{Y}^{xd} = B^b Y^{be}$. By rewinding $\mathcal{A}$ and re-programming $h(\overline{X}, Y, pid_B) = e'$, $\mathcal{S}$ can get $PS' = CDH(\overline{X}, \overline{Y}') = \overline{Y}'^{b+xd} = (BY^{e'})^{b+xd}$ with non-negligible probability in the rewound run, and consequently $CDH(B, \overline{Y}') = PS'/\overline{Y}'^{xd} = B^b Y^{be'}$. Then, $\mathcal{S}$ computes $CDH(B, Y) = (CDH(B, \overline{Y})/CDH(B, \overline{Y}'))^{(e-e')^{-1}}$, and finally $CDH(B, B) = CDH(B, \overline{Y})/CDH(B, Y)^e$, which violates the GDH assumption.

The second case, denoted Case-2, is that sid_T is run at Responder. In this case, for presentation simplicity, we denote by $U_t = pid_B$ the test-session holder, and by

$U_k = pid_A$ the peer user $peer_{sid_T}$. Let $\overline{X}||\overline{Y}$ be the session label of sid_T, where $\overline{X}$ is sent by the CMIM adversary $\mathcal{A}$ who is impersonating the honest user pid_A, and $\overline{Y}$ is sent by the uncorrupted user pid_B. Unlike the first case, $\overline{X}$ may be generated and sent by the honest user pid_A in an existing session unmatched to sid_T. The impersonation security in this case is still reduced to the AEAD security and the GDH assumption in the RO model.

We first consider the case of $pid_A \neq pid_B$ for Case-2. In this case, the GDH-solver $\mathcal{S}$ takes (Y, A) as input, where $Y, A \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(Y, A)$ with the aid of DDH-oracle. Towards this goal, it randomly guesses the peer user pid_A with successful probability $\frac{1}{n}$, generates the public-keys and secret-keys for all the honest users in the system except pid_A whose public-key is set to be A (i.e., the value given in $\mathcal{S}$'s input). Similar to the proof of Theorem 5 and as clarified above regarding the simulation for $peer_{sid_T}$ and the test-session, $\mathcal{S}$ can well simulate pid_A with the aid of DDH-oracle. In the test-session sid_T, after receiving $\overline{X}$ in the first round, it sets $\overline{Y} = BY^e$ where $e = h(\overline{X}, Y, pid_B)$ and Y is the value given in $\mathcal{S}$'s input. Then, $\mathcal{S}$ generates the accompanying AEAD ciphertext (H_B, C_B), *where the underlying keys* (K_1, K_2) *are set by* $\mathcal{S}$ *itself with the aid of the DDH-oracle to ensure the consistency of* KDF *as clarified above regarding* $peer_{sid_T}$ *simulation.* Finally, $\mathcal{S}$ sends $\{\overline{Y}, H_B, C_B\}$ in the second round of sid_T. Denote by (H_A, C_A) the AEAD ciphertext sent by the CMIM adversary $\mathcal{A}$ in the third round of sid_T, which is decrypted to be (pid_A, X) by pid_B using the key K_1 such that $\overline{X} = AX^d$ for $d = h(X, pid_A)$ as we assume sid_T is complete. Note that, in the RO model, with overwhelming probability the RO-query $d = h(X, pid_A)$ has been made before $\mathcal{A}$ sends $\overline{X}$ in the first round of sid_T; otherwise, sid_T will fail in the third round with overwhelming probability (while sid_T is assumed to be complete). Also note that the RO-query $h(X, pid_A)$ may not be made by $\mathcal{A}$ itself. For example, consider that: $\overline{X}$ and the RO-query $h(X, pid_A)$ are generated and made by pid_A in a non-matching session, where x is, however, exposed to $\mathcal{A}$.

We first observe that $(H_A, C_A) \neq (H_B, C_B)$ *even if* $pid_A = pid_B$. Otherwise, (H_A, C_A) will be decrypted into (Y, pid_B); but as Y is independent of $\overline{X}$, the probability $\overline{X} = BY^{h(Y, pid_B)}$ is negligible assuming h is RO. Note that, as we assume the test-session has no matching session, the underlying AEAD-key K_1 used in sid_T is independent of the AEAD-keys in all the other sessions. By the AEAD security in the RO model, conditioned on $\mathcal{S}$ successfully breaks the impersonation security with non-negligible probability in Case-2, $\mathcal{S}$ will get $PS = CDH(\overline{X}, \overline{Y}) = (AX^d)^{b+ye}$, from which it computes $CDH(Y, \overline{X}) = (PS/\overline{X}^b)^{e^{-1}} = A^y X^{yd}$ as we assume $pid_A \neq pid_B$ and $\mathcal{S}$ knows b. According to the general forking lemma, by rewinding $\mathcal{A}$ and re-programming $h(X, pid_A) = d'$, with also non-negligible probability $\mathcal{S}$ will get $PS' = CDH(\overline{X}', \overline{Y}') = (AX^{d'})^{b+ye'}$, where $\overline{Y}' = BY^{e'}$ is sent by $\mathcal{S}$ in the second round of the test-session (indicated by

the query $(\mathsf{Test}, U_{t_0}, U_{t_1}, \overline{X}')$ made by $\mathcal{A}$) in the rewound run. Note that, as the RO-query $h(\overline{X}, Y, pid_B)$ is posterior to $h(X, pid_A)$, rewinding and re-programming $h(X, pid_A) = d'$ causes re-programming $h(\overline{X}, Y, pid_B) = e'$. Then, $\mathcal{S}$ computes $CDH(Y, \overline{X}') = (PS'/\overline{X}'^b)^{e'^{-1}} = A^y X^{yd'}$. Finally, $\mathcal{S}$ computes $CDH(Y, X) = (CDH(Y, \overline{X})/CDH(Y, \overline{X}'))^{(d-d')^{-1}}$, and then $CDH(Y, A) = CDH(Y, \overline{X})/CDH(Y, X)^d$, which violates the GDH assumption.

Again, the above proof of Case-2 has assumed $pid_A \neq pid_B$. If $pid_A = pid_B$ where $A = B$, $\mathcal{S}$ takes input $A \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(A, A)$ with DDH-oracle. For this case, under the AEAD security in the RO model, from the KDF oracle $\mathcal{S}$ will get $PS = CDH(\overline{X}, \overline{Y}) = (AX^d)^{b+ye} = (AX^d)^{a+ye}$ for the test-session sid_T, *where* $y \leftarrow Z_q^*$ *is generated by* $\mathcal{S}$ *itself and can be exposed to adversary.* $\mathcal{S}$ can then compute $CDH(\overline{X}, A) = PS/\overline{X}^{ye} = A^a X^{ad}$. Similar to the analysis for the case of $pid_A \neq pid_B$, by rewinding $\mathcal{A}$ and re-programming $h(X, pid_A) = d'$, $\mathcal{S}$ can get $PS' = CDH(\overline{X}', \overline{Y}') = (AX^{d'})^{a+ye'}$ and compute $CDH(\overline{X}', A) = PS'/\overline{X}'^{ye'} = A^a X^{ad'}$. $\mathcal{S}$ then computes $CDH(X, A) = (CDH(\overline{X}, A)/CDH(\overline{X}', A))^{(d-d')^{-1}}$, and $CDH(A, A) = CDH(\overline{X}, A)/CDH(X, A)^d$, which violates the GDH assumption. This finishes the proof of impersonation security.

Finally, we prove the ID-SK indistinguishability. The proof of impersonation security has already established that the test-session sid_T has matching session sid'_T, where they share the same session label $\overline{X}||\overline{Y}$. The first observation is that, as clarified with the proof of ID-SK indistinguishability for UCAKE, $\overline{X}$ and $\overline{Y}$ perfectly hide players' identities in the random oracle model. Then, by the AEAD security, in order to break the ID-SK indistinguishability the CMIM adversary $\mathcal{A}$ has to make the RO-query $KDF(PS = CDH(\overline{X}, \overline{Y}), \overline{X}||\overline{Y})$ with non-negligible probability. It is then reduced to the GDH assumption in the RO model, where the proof is similar to, but actually simpler than, the proof for impersonation security. Specifically, in order to break the GDH assumption, the GDH-solver $\mathcal{S}$ does not need to rewind the adversary $\mathcal{A}$ and re-program the random oracle h, by the fact that the values $\overline{X}$ and $\overline{Y}$ are generated by the honest users in the matched sessions sid_T and sid'_T. We distinguish two cases:

- If $peer_{sid_T} = U_t$, with public-key A for presentation simplicity, the GDH-solver $\mathcal{S}$ takes A as input and its goal is to compute $CDH(A, A)$ with the oracle $DDH(A, \cdot, \cdot)$. $\mathcal{S}$ acts on behalf of $peer_{sid_T}$, by honestly generating $X = g^x$ (resp., $Y = g^y$) in sid_T (resp., sid'_T) and using its DDH-oracle to ensure KDF consistency. After getting $PS = CDH(\overline{X}, \overline{Y}) = (AX^d)^{a+ye}$, $\mathcal{S}$ can recover $CDH(A, A)$ with (x, y) known to it.
- If $peer_{sid_T} \neq U_t$, where one of $\{peer_{sid_T}, U_t\}$ denotes pid_A and the other denotes pid_B for presentation simplicity. The GDH-solver $\mathcal{S}$ takes as input (U, V), where

$U, V \leftarrow G \setminus 1_G$, and its goal is to compute $CDH(U, V)$ with DDH-oracle. The public-key and secret-key of U_t, as well as the DH-component to be sent by $peer_{sid_T}$ in sid'_T, are generated by $\mathcal{S}$ itself, while the public-key of $peer_{sid_T}$ and the DH-component to be sent by U_t in sid_T are set to be (U, V), i.e., the input given to $\mathcal{S}$. From $PS = CDH(\overline{X}, \overline{Y}) = (AX^d)^{b+ye}$ and because $\mathcal{S}$ knows either (a, y) or (b, x), $\mathcal{S}$ can compute $CDH(U, V)$ that is either $CDH(B, X)$ or $CDH(A, Y)$, which violates the GDH assumption. □

Corollary 3. *Let $\mathcal{X}$ (resp., $\mathcal{Y}$) be a well-spread distribution over Z_q^* with min-entropy $\min(\lambda_{\mathcal{X}}, \lambda_{\mathcal{Y}}) > \omega(\log |q|)$. The protocol described in Fig. 5, when x (resp., y) is taken according to $\mathcal{X}$ (resp., $\mathcal{Y}$), is strongly CAKE-secure, under the AEAD security and the FGDH and FGDL assumptions in the random oracle model.*

10.2 Full TLS Handshake: The Third Solution

In this section, based upon CAKE and higncrption, we present a full-fledged solution for TLS handshake, combining 0-RTT mode, 0.5-RTT mode, uniliteral authentication and mutual authentication. The basic protocol structure of the full-fledged TLS handshake is presented in Fig. 8, where symbols are inherited from Fig. 6 and Fig. 16.

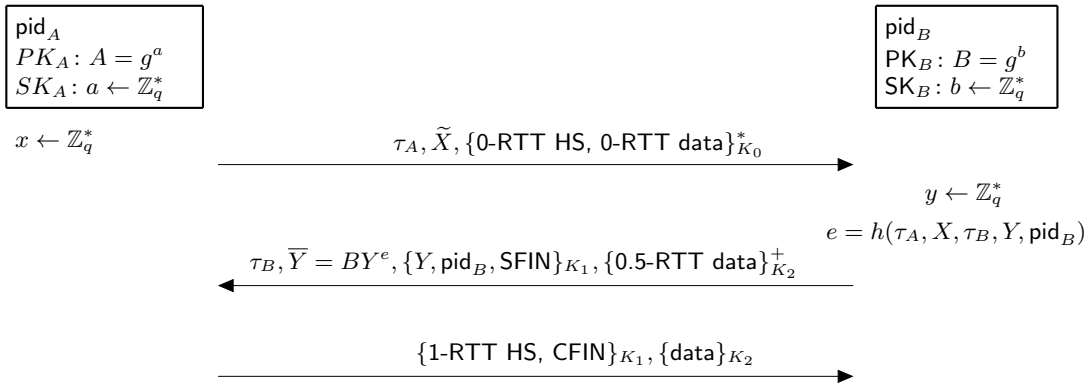

Fig. 8: Full-fledged TLS handshake based on CAKE and higncryption

As in CAKE, at one's own discretion (according to task criticality and application scenarios), user pid_A (resp., pid_B) can select $x \leftarrow \{0,1\}^{l_A}$ (resp., $y \leftarrow \{0,1\}^{l_B}$), where $\omega(\log |q|) < l_A, l_B \leq |q|$, and l_A and l_B may not necessarily be equal. For example, user pid_A can set, according to task criticality and application scenarios, $l_A = \lceil |q|/4 \rceil$, or $l_A = \lceil |q|/2 \rceil$, or $l_A = \lceil 3|q|/4 \rceil$, or just $l_A = \lceil 3|q|/4 \rceil$. In practice, it may be recommended to set $l_A = l_B = \lceil |q|/2 \rceil + 1$.

In the protocol description, according to the modes of protocol run, some messages may have different forms. *However, with CAKE and higncryption, all of these different modes of protocol run are well unified within a single protocol structure in a modular and flexible way.*

- For mutual authentication with 0-RTT, $\widetilde{X} = AX^d$ for $d = h(\tau_A, X, pid_A, pid_B, pss)$, where the pre-shared state pss is empty if client and server didn't pre-share some state before protocol run (which is mainly for TLS run in the PSK mode). In this case, 0-RTT HS contains (X, pid_A) while 1-RTT HS is empty, where the first round constitutes a higncryption scheme.
- For mutual authentication without 0-RTT, $\widetilde{X} = AX^d$ for $d = h(\tau_A, X, pid_A, pid_B, pss)$ and 1-RTT HS contains (X, pid_A). This case actually corresponds to CAKE.
- For unilateral authentication mode, $\widetilde{X} = g^x$, and both 0-RTT HS and 1-RTT HS are empty.

The key K_0 is derived from $CDH(\widetilde{X}, B)$ and public transcript including (pid_A, pid_B, X) and even pre-shared state if it is run in the resumption mode. The keys K_1 and K_2 are derived from $PS = CDH(\widetilde{X}, \overline{Y})$ (and public transcript and possibly pre-shared state), in a way that K_1 and K_2 are computationally independent as is done in TLS1.3 or QUIC.[21]

References

1. J. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. *EUROCRYPT* 2002: 83 – 107.
2. J. Baek, R. Steinfeld, and Y. Zheng. Formal Proofs for the Security of Signcryption. *Journal of Cryptology* (2007) 20: 203-235.
3. M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *ASIACRYPT* 2000: 531-545.
4. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma, *ACM CCS*, 2006: 390-399.
5. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *CRYPTO* 1993: 273 – 289.
6. D. Boneh. The Decision Diffie-Hellman Problem. *ANTS* 1998: 48-63.
7. C. Brzuska, M. Fischlin, B. Warinschi, and S.C. Williams. Composability of Bellare-Rogaway key exchange protocols. *ACM CCS* 2011: 51 – 62. 2011.
8. C. Brzuska, N.P. Smart, B. Warinschi, G.J. Watson. An Analysis of the EMV Channel Establishment Protocol. *ACM CCS* 2013: 373-386.
9. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html
10. R. Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. *CRYPTO* 1997: 455-469.
11. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. *EUROCRYPT* 2001: 453-474.

[21] If 0.5-RTT authentication is not run, the session-key K_2 can be derived from PS and (the hash of) the transcript up to $\{\mathsf{data}\}_{K_2}$ or $K_2 = KDF(PS, X||pid_A||Y, pid_B)$.

12. R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. *CRYPTO* 2002: 143-161.
13. A.W. Dent. Hybrid Cryptography. Cryptology ePrint Archive, Report No. 2004/210.
14. Y. Dodis and J.H. An. Concealment and Its Applications to Authenticated Encryption. *EUROCRYPT* 2003: 312-329.
15. B. Dowling, M. Fischlin, F. Günther and D. Stebila. A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates. *ACM CCS* 2015: 1197-1210.
16. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, Draft-12, 2016. https://tools.ietf.org/html/draft-ietf-tls-tls13-12
17. J. Fan, Y. Zheng, and X. Tang. A Single Key Pair is Adequate for the Zheng Signcryption. *ACISP* 2011: 371-388.
18. M.Fischlin and F. Günther. Multi-Stage Key Exchange and the Case of Google' s QUIC Protocol. *ACM CCS* 2014: 1193-1204.
19. GnuTLS Documentation. http://www.gnu.org/software/gnutls/ documentation.html, 2011.
20. M. Gorantla, C. Boyd, J. Gonzalez Nieto. On the Connection Between Signcryption and One-Pass Key Establishment. *Cryptography and Coding* 2007: 277-301.
21. S. Halevi and H. Krawczyk. One-pass HMQV and asymmetric key-wrapping. *PKC* 2011: 317-334.
22. T. Jager and J. Schwenk. On the Equivalence of Generic Group Models. *ProvSec* 2008: 200-209.
23. T. Jager, F. Kohlar, S. Schäge and J. Schwenk. On the Security of TLS-DHE in the Standard Model. *CRYPTO* 2012: 273-293.
24. H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). *CRYPTO* 2001: 310-331.
25. H. Krawczyk. SIGMA: The 'sign-and-mac' approach to authenticated Diffie-Hellman and its use in the IKE-protocols. In *CRYPTO*, pages 400-425, 2003.
26. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO*, pages 546–566, 2005.
27. H. Krawczyk, K.G. Paterson and H. Wee. On the Security of the TLS Protocol: A Systematic Analysis. *CRYPTO* 2013: 429-448.
28. Hugo Krawczyk and Hoeteck Wee. The OPTLS Protocol and TLS 1.3. *EuroS&P 2016*.
29. B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16, 2007.
30. R. Lychev, S. Jeroy, A. Boldyrevaz and C. Nita-Rotarux. How Secure and Quick is QUIC? Provable Security and Performance Analyses. *IEEE Security & Privacy* 2015: 214-231.
31. U. Maurer. Abstract Models of Computation in Cryptography. *IMA Cryptography and Coding* 2005: 1-12.
32. U. Maurer and S. Wolf. Diffie-Hellman Oracles. *CRYPTO* 1996: 268-282.
33. U. Maurer and S. Wolf. Lower Bounds on Generic Algorithms in Groups. *EUROCRYPT* 1998: 72-84.
34. D. A. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. *INDOCRYPT* 2004: 343-355.
35. A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *SAC*, pages 70–88, 1995.
36. T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. *PKC* 2001: 104-118.
37. K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. *ASIACRYPT* 2011: 372-389.
38. P. Rogaway. Authenticated-Encryption with Associated-Data. *ACM CCS* 2002: 98-107.
39. P. Rogaway and T. Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. *EUROCRYPT* 2006: 373-390.
40. J. Roskind. Quick UDP Internet Connections: Multiplexed Stream Transport over UDP. 2012.
41. C. P. Schnorr. Small Generic Hardcore Subsets for the Discrete Logarithm. *Information processing Letters* 79(2): 93-98, 2001.
42. J. T. Schwartz. Fast Probabilistic Algorithms for Verifications of Polynomial Identities. *Journal of the ACM*, 27(3): 701-717, 1980.

43. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. *EUROCRYPT* 1997: 256-266.
44. C. Wright, S. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. *NDSS* 2009.
45. A. C. Yao and Y. Zhao. Deniable Internet Key-Exchange. *ACNS* 2010: 329-348.
46. A. C. Yao and Y. Zhao. OAKE: A New Family of Implicitly Authenticated Diffie-Hellman Protocols. *ACM CCS* 2013: 1113-1128.
47. A. C. Yao and Y. Zhao. Privacy-Preserving Authenticated Key-Exchange Over Internet. *IEEE Transactions on Information Forensics and Security*, 9(1): 125-140, 2014. Preliminary version appears at Cryptology ePrint Archive, Report No. 2011/035, 2011.
48. Y. Zheng. Digital signcryption or how to achieve cost(Signature & encryption) $\ll$ cost(Signature)+cost(Encryption). *CRYPTO* 1997: 165-179.

A Related Protocols

A.1 Zheng's Signcryption

Zheng's signcrypton is briefly described in Fig. 9, where $H : \{0,1\}^* \rightarrow Z_q$ is a hash function.

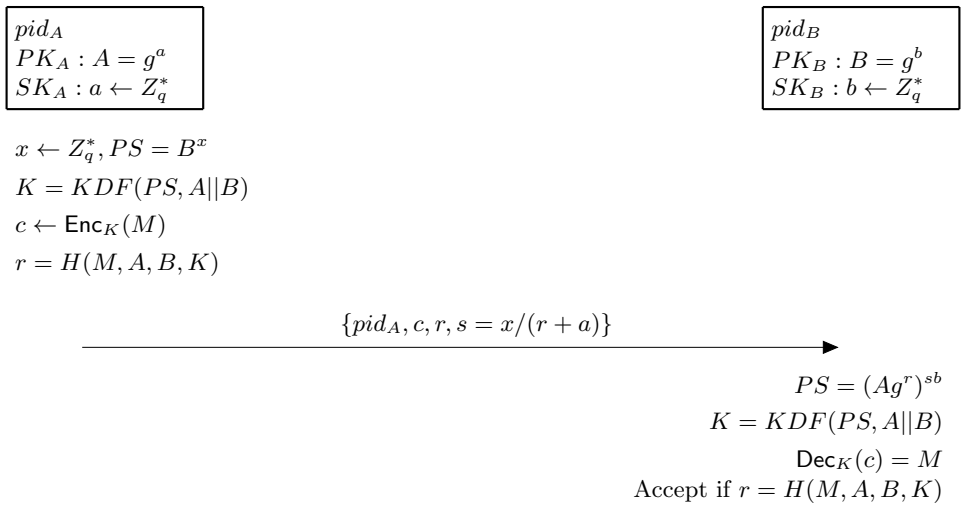

Fig. 9: Protocol structure of Zheng's signcryption

A.2 One-Pass HMQV (HOMQV)

The brief protocol structure of HOMQV, with sender authentication, is described in Fig. 10, where the session-key is set to be K_2.

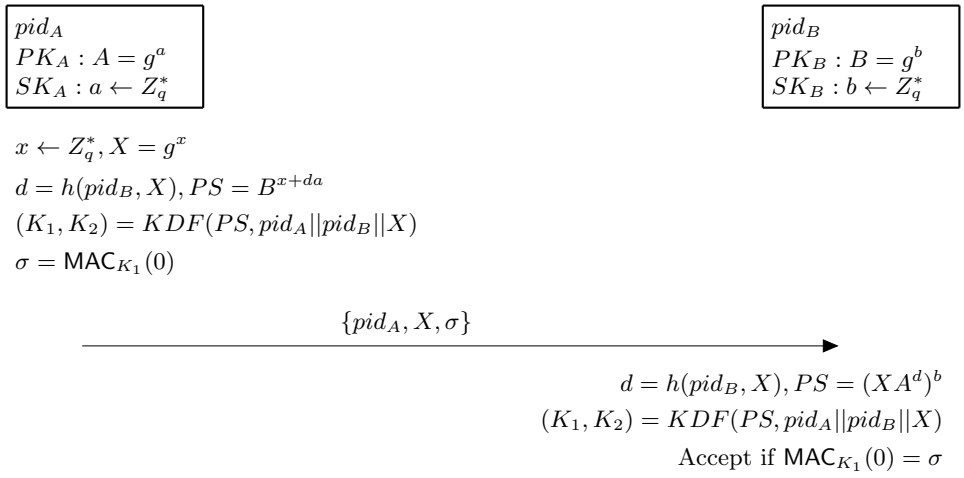

Fig. 10: Protocol structure of HOMQV

A.3 Basic Mechanism of QUIC

The basic protocol structure of QUIC is presented in Fig. 11. The client is assumed to have already possessed the public identity information of the server, as well as a state information stk (that is an AEAD encryption of client's IP-address and server's time-stamp) generated by the server in the initial connection. K_1 is derived from $CDH(X, B)$ and some auxiliary input (determined by the public transcript and stk), the session-key K_2 is derived from $CDH(X, Y)$ and some auxiliary input. "$\{\ \}_K$" denotes AEAD encryption using key K, and "$\{\ \}_K^+$" means it is optionally generated and sent only for 0-RTT mode. In actual protocol implementation of QUIC, the client and server also generate and exchange some auxiliary public information (like random nonces, client session identifier, etc), which are omitted in the above basic protocol structure.

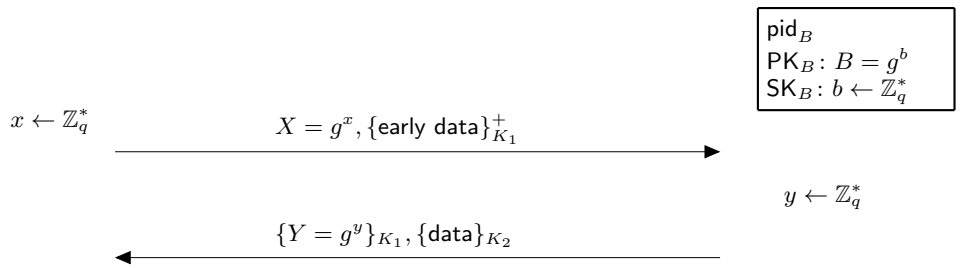

Fig. 11: Basic structure of QUIC

A.4 Basic Mechanism of TLS1.3

The basic authentication mechanism of TLS1.3 is presented in Fig. 12, which is actually based on SIGMA [25]. σ is server's signature on the hash of public transcript, K_1 used for encrypting (pid_B, σ) and the session-key K_2 are derived from $CDH(X, Y)$ in a way that K_1 and K_2 are computationally independent.

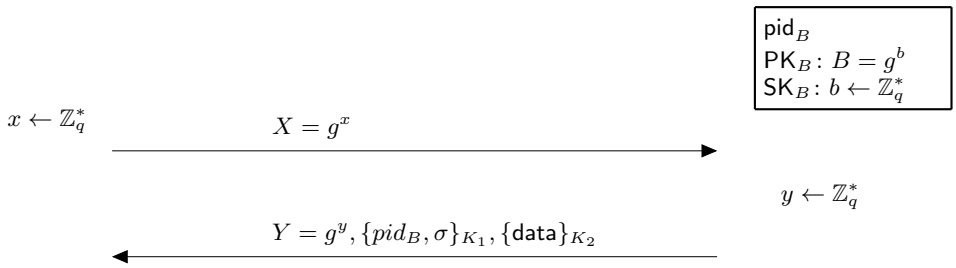

Fig. 12: Basic authentication mechanism of TLS1.3

A.5 Basic Mechanism of Deniable IKE (DIKE)

The basic mechanism of DIKE, proposed and implied in [45,47], is presented in Fig. 13, where K_2 is derived from both g^{xb} and g^{xy}, K_1 and the session-key K are derived from g^{xy} in a way that K_2 and K are computationally independent. In the actual protocol description in [47,45], $MAC_{K_2}(X, Y, pid_B)$ is written as $H(X^b, X^y, X, Y, pid_B)$. The works [45,47] put focus on deniability and authentication. It is shown that the session transcript of DIKE, together with the resultant session-key, is forwardly deniable. In addition, deriving the session-key K merely from g^{xy} (and some auxiliary input determined from public transcript and shared state) has the following advantages: (1) deniability, i.e., the session-key cannot be served as a witness to session participance; (2) it renders the receiver a way to quickly recover the encrypted data data, without having to first decrypt c_1 and check σ which can be performed in parallel with (even posterior to) decrypting c_2.

A.6 Basic Mechanism of OPTLS

The basic mechanism of OPTLS is given in Fig. 14, where , K_1 is derived from g^{xy}, K_2 is derived from g^{xb}, and K is derived from both g^{xb} and g^{xy}.

A.7 Basic Structure of (H)MQV and OAKE

The basic protocol structures of (H)MQV and OAKE are given in Fig. 15.

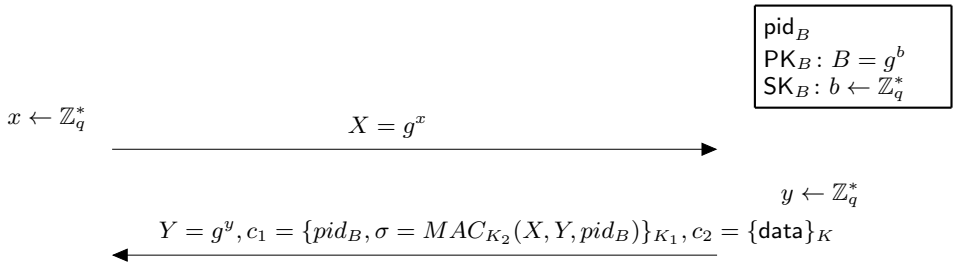

Fig. 13: Basic Structure of DIKE

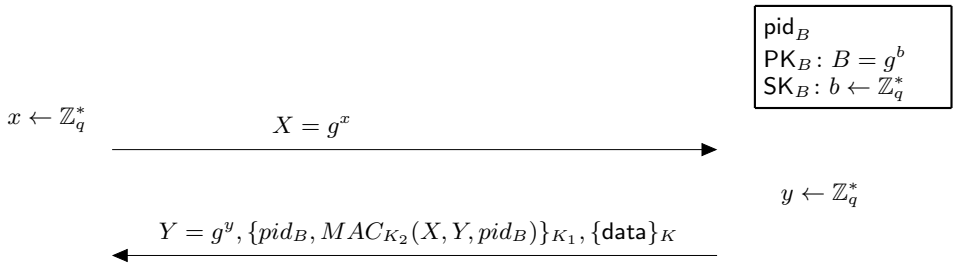

Fig. 14: Basic Structure of OPTLS

B Generalized Protocol Structure of CAKE

The generalized protocol structure of CAKE is depicted in Fig. 16 (page 63). There, τ_A and τ_B denote some auxiliary information (like random nonces, protocol version, extensions, system parameters, timing stamps,etc) sent by pid_A and pid_B respectively; m_A (resp., m_B) denotes the set of extra messages to be encrypted by user pid_A (resp., pid_B). In practice, m_B should only contain non-critical messages (e.g., a time-stamp) and could be encrypted with K_2 as in 0.5-RTT mode of TLS-1.3, as the peer's (say, pid_A's) authentication has not been established at the moment of sending C_B. On the other hand, user pid_A can safely include, to its wish, any message in m_A. The value aux_A (resp., aux_B) represents the set of some auxiliary information taken into the input of d (resp., e), where pss denotes some pre-shared state between client and server (e.g., the pre-shared key in the resumption mode of TLS-1.3).

The generalized protocol structure of CAKE allows flexible implementations and deployments in practice. In the basic version of CAKE, on which provable security is to be conducted, $aux_A = aux_B = m_A = m_B = \emptyset$ (i.e., all of them are the empty set $\emptyset$), which shows the provable security of CAKE does not depend upon these values.

On the other hand, for a more robust version in practice, some of the following implementations may come into operation, by the agreement of both parties and according to the application scenarios.

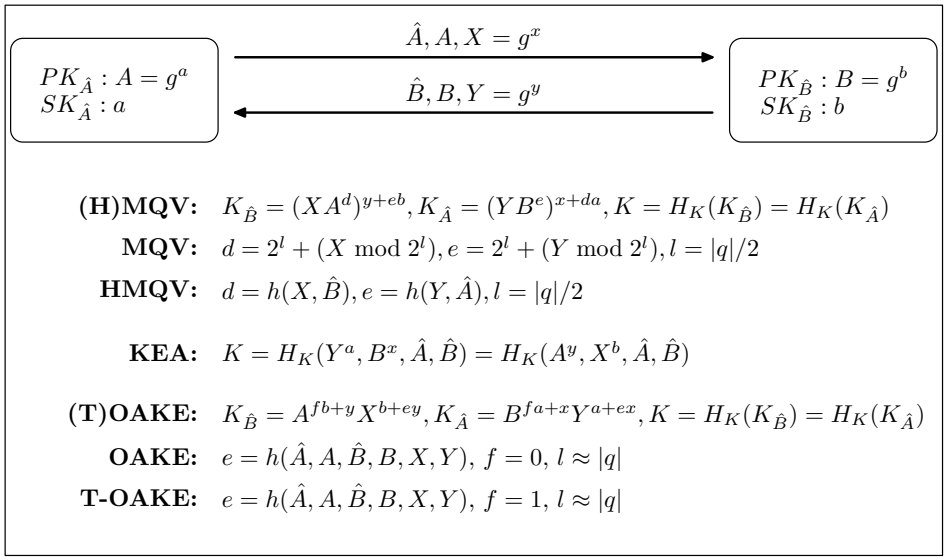

Fig. 15: Specifications of (H)MQV, KEA and (T)OAKE

– At the side of pid_A, let aux_A include a time-stamp $t_A \in m_A$ (in this case the user pid_B will check the validity of t_A after decryption of C_A), and pid_B (and some other information from the server configuration like ciphersuits, parameters, server configuration identifier, etc) in case it is known to user pid_A before the protocol starts, and some pre-shared state pss (e.g., when the protocol is run in the resumption mode). In some application scenarios, aux_A can also contain a random nonce sent by the responder in an additional prior round.
– At the side of pid_B, let aux_B include $\overline{X}$, and/or a time-stamp $t_B \in m_B$ (in this case the user pid_A will check the validity of t_B after decryption of C_B).

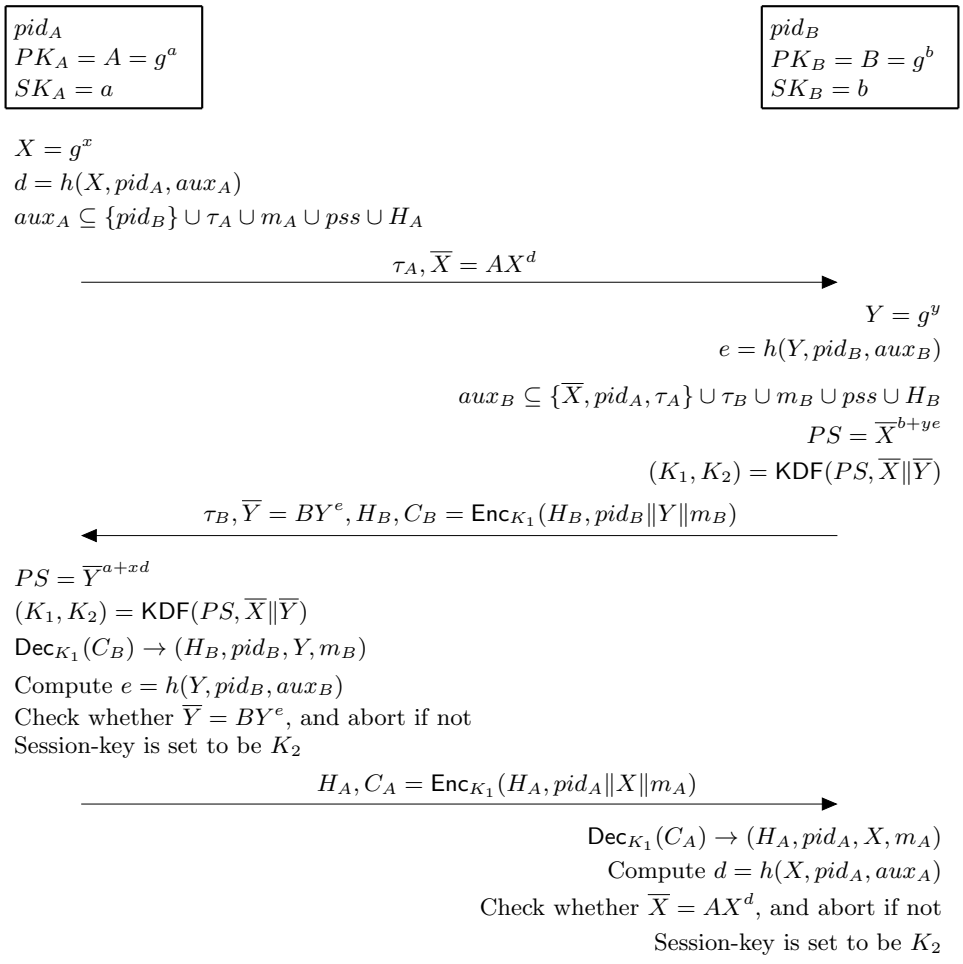

Fig. 16: General protocol structure of CAKE