Identity-Concealed Authenticated Encryption and Key Exchange[*]

Yunlei Zhao
Shanghai Key Laboratory of Data Science
Software School, Fudan University, China
State Key Laboratory of Cryptology, Beijing, China
ylzhao@fudan.edu.cn

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 then 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). 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) *connection* are two of current research focuses in the design and analysis of cryptographic systems (in particular, 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 [13], QUIC [30], EMV [6]. 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 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 to TLS [24]. According to Google's measurement, currently at least 75% of all QUIC connections use 0-RTT mode. Unfortunately, QUIC now only supports 0-RTT mode *without* client authentication. It is suggested that, in order for QUIC to be ubiquitous, a suitable mechanism for 0-RTT client authentication is needed.

To our knowledge, the literature lacks a cryptographic mechanism that solidly and practically integrates public-key encryption, entity authentication and ID-concealment into a monolithic primitive. A natural solution to identity-concealed 0-RTT client authentication is to encrypt client's 0-RTT data and signature using server's public-key [23]. However, this approach has several drawbacks and may not be satisfactory enough.

- Firstly, composing public-key encryption and digital signature *serially* 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 on authenticated encryption (AE) even though AE can be generally achieved by composing CPA-secure encryption and MAC.
- Secondly, such a serial composition may not be sound enough and may bring some security concerns [34, 33].
- 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

*This research was supported in part by NSFC (Grant Nos. 61472084, 61272012, U1536205, 61332019) and 973 Grant No. 2014CB340601 and Shanghai Innovation Action Project No. 16DZ1100200.

CCS'16, October 24 - 28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4139-4/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2976749.2978350

authenticated encryption may not be so obvious and deserves explorations, as already witnessed by the modeling of composition of encryption and authentication [1, 2, 11, 16, 17].

- 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 [37], which is also standardized by ISO-29150. It is shown that signcryption is functionally equivalent to one-pass authenticated key-exchange [10, 16], which in turn has applications in asymmetrical key-wrapping [17]. However, for Zheng's signcryption [37, 2, 14] and one-pass HMQV (HOMQV) [20, 17], sender's public identity information (including its certificate) has to be sent in clear, or otherwise the receiver cannot derive the shared key. It would be interesting to note that, although signcryption (1997) [37] and one-pass MQV (1995) [27] have been proposed 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 σ corresponding to sender's signature cannot be generated by the receiver. In addition, Zheng's signcryption suffers from the x-security defined in [17]:[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 motivating question.

Motivating Question 1

Can we come up with a cryptographic mechanism that satisfies: (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, and is the backbone of a list of network security protocols that are widely standardized and in use. Up to now, the most efficient AKE protocols are (H)MQV [27, 20] and OAKE [36], but none of them considers ID-concealment. In the client/server setting, TLS1.3 and QUIC represent the most efficient secure channel establishment protocols, both of which support ID-concealment. The protocol structure of QUIC, briefly described in Appendix B, is conceptually simple and enjoys the advantages of efficiency, receiver deniability, and deployment flexibility. However, QUIC (even 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 server's DH-component Y, and consequently server's ID-privacy is lost. The basic authentication mechanism of TLS1.3 is based on the SIGMA scheme [19], which is also briefly described in Appendix B. For presentation simplicity, by TLS1.3 we mean the core authentication mechanism presented there. TLS1.3 uses signature for server authentication, which has the following effects. On the one hand, when implemented with EC-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). Since its sixth version, TLS1.3 incorporates the OPTLS protocol [22], which is also described in Appendix B. Briefly speaking, OPTLS is as efficient as QUIC, while enjoys forward ID-privacy. But none of QUIC, TLS1.3 and OPTLS enjoys x-security, i.e., the exposure of DH-exponent in a session will expose the session-key (for QUIC and TLS1.3) or server's ID (for OPTLS). This leads to the following motivating question.

Motivating Question 2

Can we come up with new AKE schemes, which satisfy simultaneously: (1) forward ID-privacy, (2) better efficiency than QUIC and OPTLS, (3) x-security, (4) receiver deniability, and (5) being free of signatures?

We note that existing security definition frameworks for AKE or secure channel establishment are not well suitable for identity-concealed AKE (CAKE). Traditional CK-framework [8] and its variants (e.g., the post-ID CK-framework [9]) are not applicable to the analysis of CAKE. Session matching, which is at the heart of defining AKE security in CK-framework, critically relies on the fact that players' identities and public-keys are exchanged explicitly, and thus can be publicly verified according to session transcripts alone. Public verification of session matching is, in turn, crucial for provable composition with subsequent symmetric-key cryptographic primitives [8, 5].

In the BR-framework [4], session matching is defined w.r.t. two sessions of the identical session transcript. 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). Such a requirement appears to be too strict, which causes limited applicability or more complex analysis, as the following examples demonstrated: (1) 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; (2) Consider AKE protocols, e.g., the TLS handshake, where some protocol messages are encrypted by a probabilistic symmetric encryption scheme using keys derived from some preliminary shared key (PSK). For such protocols, an adversary could get PSK in one session (e.g., by a state reveal query), and then use PSK to re-encrypt plaintext messages of the exposed session in another session. This can also cause two sessions of different transcripts to have the same session-key; (3) For AKE protocols deployed in a complex or critical system, we may want them to enjoy some capability of fault tolerance, in the sense that the loss or modification of some values (e.g., caused by transmission in a poor network) could be tolerated or recovered. However, such an approach is not well supported by the BR-framework.

Identity privacy for secure transport protocols was also formulated in some existing works (e.g., [6]). But ID-privacy was treated separately from the security definition for authenticated key-exchange or channel establishment.

[1]This is actually named as y-security in [17], where the player pid_B plays the role of sender.

Recently, a new security definitional framework for *multi-stage* key-exchange (MSKE) protocols was introduced [15, 12, 5], whose power is proven in analyzing secure transport protocols like QUIC [15] and TLS1.3 [12]. We notice that the MSKE framework is not well applicable to CAKE either, on the following grounds.

Firstly, ID-privacy was not considered in the MSKE framework [15, 12, 5]. 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 [15, 12, 5]. This means that we have to divide the (actually single) session run of a CAKE protocol into several stages, such that the partial session run upon agreeing k_{id} should be treated as one separate stage [15, 12]. In the MSKE framework, it is assumed that during the session run, once a stage is finished, the protocol execution is immediately suspended and the control is returned to the adversary. This could lead to several negative effects, as discussed below.

- The way of dividing a single CAKE protocol run into multiple stages, and suspending and giving control to adversary immediately upon completion of each stage, 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 a subsequent stage.
- It may result in unnatural modeling of secrecy exposure. Specifically, the intermediate key k_{id} is usually transient, particularly compared with some offline computed and stored values like DH-exponents and static secret-keys. If we grant the adversary the ability to expose so temporal secrecy like k_{id}, no security guarantee could 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 [15, 12], 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 [15, 12], the adversary is not allowed to adaptive register users, has limited capability of secrecy exposure (where exposing DH-exponent is denied, and static secret-key exposure is not distinguished from user corruption), and is required to indicate the authentication type upon session initiation.

As a consequence, it would be much desirable to develop a new definitional framework for CAKE, which enjoys the following advantages simultaneously. (1) *Composability*: it verifies session matching in public, thus salvaging the composability; (2) *Conciseness*: it integrates both AKE security and identity privacy, simplifying security definition and analysis; (3) *Unification*: it unifies the dominant frameworks of BR, CK and post-ID CK; (4) *Robustness* and *versatility*: it allows powerful CMIM adversary with adaptive party registration and strong capability of secrecy exposure, and implies 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 traditional CK or BR frameworks.

Motivating Question 3

Can we come up with a new definitional framework for *identity-concealed* AKE, which enjoys simultaneously: (1) public checkability of session matching, (2) conciseness (3) unification, and (4) robustness and versatility?

1.1 Contributions

In this work, we systematically solve the above three motivating questions. For the first motivating question, we introduce a new primitive, referred to as *h*iding-*i*dentity si*gncryption* (*higncryption*, for short), which integrates public-key encryption, digital signature and ID-concealment into a monolithic primitive in a solid and practical way. We then present the security definitional framework for higncryption, and a conceptually simple (yet carefully designed) construction of higncryption, with detailed comparisons with Zheng's signcryption and HOMQV.

As a new primitive, higncryption may be of independent value and can have many applications. A direct application of higncryption is one-pass ID-concealed authenticated key-exchange protocol, which can, in turn, be applied to key-wrapping. We make in-depth discussions on its applications to 0-RTT authentication, showing higncryption is well suitable to (and compatible with) QUIC and OPTLS. Then, we discuss the implication of higncryption to (*identity-concealed*) AKE (CAKE) server-only authenticated unilateral CAKE (UCAKE), with comparisons with QUIC, OPTLS and TLS1.3 in accordance with the second motivating question.

For the third motivating question, a new concise security definitional framework for CAKE and UCAKE is proposed in this work, which unifies the traditional BR and (post-ID) frameworks, enjoys public checkability of session matching, and ensures very strong security guarantee.

All the protocols developed in this work are provably secure under standard assumptions in the random oracle (RO) 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. We study the complexity of FDL and related problems in the generic group model, showing their intractability hold in the generic group model, which 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 $V \leftarrow Alg$ to mean that algorithm Alg runs and outputs value named V. 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 $\rho = N/q$ 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 ρ is typically very small.

The discrete logarithm (DL) assumption over G says that given $X = g^x$, where $x \leftarrow Z_q^*$, no probabilistic polynomial-time (PPT) DL-solver algorithm can output x with non-negligible probability. The computational Diffie-Hellman (CDH) assumption says that given $X = g^x$, $Y = g^y$, where $x, y \leftarrow Z_q^*$, no probabilistic polynomial-time CDH-solver algorithm can compute $CDH(X,Y) = g^{xy}$ with non-negligible probability. The Gap Diffie-Hellman (GDH) assumption 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 $Z = CDH(U, V)$.

Authenticated encryption. 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) into a ciphertext C in such a way that C provides both privacy (of M) and authenticity (of C and H) [29]. 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).

Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme. The probabilistic polynomial-time algorithm K_{se} takes the 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 $\mathsf{Enc} : \mathcal{K} \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ and the (deterministic) polynomial-time decryption algorithm $\mathsf{Enc} : \mathcal{K} \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ 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 the security game for AEAD. We define the advantage of A to be $\mathbf{Adv}_{\mathsf{SE}}^{\text{aead}}(A) = \left|2 \cdot \Pr[\text{AEAD}_{\mathsf{SE}}^A \textit{ returns } \mathsf{true}] - 1\right|$. We say that the SE scheme is AEAD-secure, if for any sufficiently large κ the advantage of any PPT adversary is negligible.

The above AEAD security is quite strong. In particular, it means that, after adaptively seeing a polynomial number of ciphertexts, an efficient adversary is unable 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 [28].

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

Table 1: AEAD security game

3. STRONG SECURITY MODEL FOR HIGNCRYPTION

A *higncryption* scheme $\mathcal{HC}$, with associated data, is specified by four polynomial-time algorithms: setup, keygen, $\mathsf{higncrypt}$ and $\mathsf{unhigncrypt}$.

setup: 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*.

$\mathsf{key\text{-}gen}$: a PPT algorithm that takes *params* as input, and outputs a public-private key pair (pk, sk) used for higncryption and unhigncryption. For presentation simplicity, we assume *params* is included in pk.

$\mathsf{higncrypt}$: 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 higncrypted. It returns a higncryptext C, or the symbol $\perp$ indicating higncryption failure. The associated data H, if any, appears in clear in the higncryptext $C \neq \perp$. *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., the DH-exponent used in generating the higncryptext C, is specified and stored in a variable $\mathcal{ST}_C$ that could be exposed to allow for a more robust security definition.

$\mathsf{unhigncrypt}$: 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 higncryptext C. It outputs either (pid_s, M) or an error symbol $\perp$. *Note that, unhigncrypt does not take sender's public identity information* pid_s *as input.*

The *correctness* for higncryption can be trivially defined. Below, we focus on the *strong* security model for higncryption in the multi-user setting, where *each user possesses a single key pair for both higncryption and unhigncrypton and can higncrypt 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., $\mathsf{DISHONEST}$) 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 $\mathsf{Corrupt}$ 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 $C = \mathsf{higncrypt}(sk_s, pid_s, pid_r, H, M))$ if $pid_s \in \mathsf{HONEST}$; Otherwise, it 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), it returns $\mathsf{unhigncrypt}(sk_r, pid_r, C))$ if $pid_r \in \mathsf{HONEST}$; Otherwise, it 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 signcryption, 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.

$\mathsf{Corrupt}$: 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 (OU). The goal of an OU-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 $\mathsf{Corrupt}$ 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 that the following hold simultaneously:

– $\mathsf{unhigncrypt}(sk_{r^*}, pid_{r^*}, C^*) = (pid_{s^*}, M^*)$, 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 the 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. In 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 models of signcryption. Also, 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 any sufficiently large security parameter. The definition of *inside unforgeability* is almost the same as that of outside unforgeability, except that oracle query $\mathsf{Corrupt}(pid_{r^*})$ is allowed to the adversary.

Insider confidentiality. The goal of an insider confidentiality adversary $\mathcal{A}^{IC}$ is to break the confidentiality of the message or 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 a pair of equal-length messages (M_0, M_1), an associated data H^*, and two pairs of $(pid_{s_0^*}, pid_{r^*})$ and $(pid_{s_1^*}, pid_{r^*})$ of equal lengths, where $pid_{s_0^*}, pid_{s_1^*}, pid_{r^*} \in \mathsf{HONEST}$, and submits them to the challenger. The challenger takes $\sigma \leftarrow \{0,1\}$, and gives $\mathcal{A}^{IC}$ the challenge higncryptext $C^* = higncrypt(sk_{s_\sigma^*}, pid_{s_\sigma^*}, pid_{r^*}, H^*, M_\sigma)$. (For simplicity, we do not model receiver ID-privacy, which can be extended by indicating a pair of challenge receivers.)
– *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', as the guess of the random 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 any sufficiently large security parameter, its advantage defined below is negligible:

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

4. CONSTRUCTION OF HIGNCRYPTION

Our starting point is the underlying mechanism, namely *non-malleable joint proof-of-knowledge* (NMJPOK), for OAKE [36] as follows. Given a challenge $Y = g^y$, the prover with static $pid_A = \{id_A, A = g^a, cert_A\}$ and ephemeral $X = g^x$ can non-malleably prove its joint knowledge of (a, x), by sending $\{pid_A, X, NMJPOK(a, x) = Y^{a+xd}\}$ where $d = h(X, pid_A, Y)$. To hide user identity, a new mechanism, referred to as *concealed non-malleable joint proof-of-knowledge* (CNJPOK), is introduced, which can be viewed as the dual of NMJPOK. With CNJPOK, the prover sends $\{\overline{X} = AX^d, C = AE_K(pid_A, X)\}$, where K is derived from $CHD(\overline{X}, Y)$. This directly leads to a two-round UCAKE protocol. Our higncryption is built upon CNJPOK, by replacing Y with receiver's public identity information.

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

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 of secret key during public key registration, but it performs sub-group membership check for each registered public key, i.e., checking $pk_i \in G \setminus 1_G$. In this work, *we assume users' identities and public-key information to be of equal lengths*; Otherwise, we need *stateful length-hiding authenticated encryption* (SLHAE) as defined in [28, 21].

Higncryption. Let $\mathsf{SE} = (\mathsf{K}_{se}, \mathsf{Enc}, \mathsf{Dec})$ be an AEAD scheme, $h : \{0,1\}^* \rightarrow \{0,1\}^l \cap Z_q^*$ 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, where $\mathcal{K}$ is the key space of K_{se}. 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. (1) Take $x \leftarrow Z_q^*$, 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.* (2) Compute $\overline{X} = AX' = AX^d$,[2] and pre-shared secrecy $PS = CDH(\overline{X}, B) = B^{a+xd} \in G$. (3) Derive keys $(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$, where $K_1 \in \mathcal{K}$, K_2 is empty for higncryption, or $K_2 \in \mathcal{K}$ for one-pass identity-concealed AKE and in this case the joint distribution of (K_1, K_2) is computationally indistinguishable from uniform distribution over $\mathcal{K} \times \mathcal{K}$. (4) Compute $C_{AE} \leftarrow \mathsf{Enc}_{K_1}(H, pid_A||X||M)$. Notice that the DH-component X is sent being encrypted. (5) Send the higncryptext $C = (H, \overline{X}, C_{AE})$ to receiver.

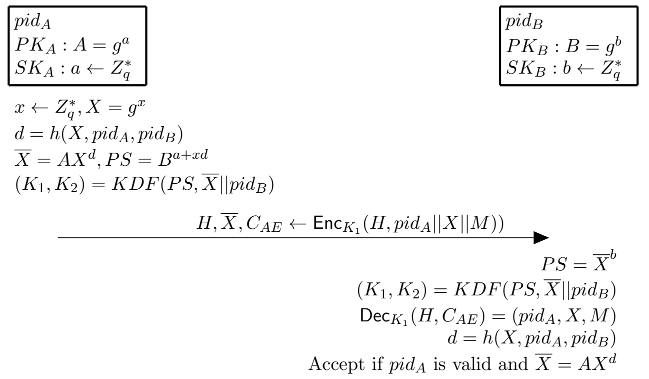
Unhigncryption. After receiving $C = (H, \overline{X}, C_{AE})$, $\mathsf{unhigncrypt}(sk_B = b, pid_B, C)$ works as follows. (1) Compute the pre-shared secrecy $PS = CDH(B, \overline{X}) = \overline{X}^b$, and derive the keys $(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$. (2) Run $\mathsf{Dec}_{K_1}(H, C_{AE})$. If $\mathsf{Dec}_{K_1}(H, C_{AE})$ returns $\perp$, abort; otherwise, get $\{pid_A = (id_A, A, cert_A), X, M)\}$. (3) 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 identity-concealed AKE is also presented in Figure 1.

4.1 Discussion and Clarifications

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'. Note that, if the cofactor ρ is small, the subgroup test of $\overline{X}$ can be essentially reduced to check $\overline{X} \in G'$ and $\overline{X}^\rho \neq 1_G$, which guarantees $\overline{X}$ is not in a small subgroup of G' of the order being a factor of ρ (though it may not fully ensure $\overline{X} \in G$). In practice, we recommend the following variant with embedded subgroup test, where

[2]An alternative way is to set $\overline{X} = A^dX$, which does not sacrifice provable security. We prefer to setting $\overline{X} = AX^d$, for the reason that, as we shall see in Section 6, it allows more flexible and efficient implementations.



pid_A
$PK_A : A = g^a$
$SK_A : a \leftarrow Z_q^*$
pid_B
$PK_B : B = g^b$
$SK_B : b \leftarrow Z_q^*$
$x \leftarrow Z_q^*, X = g^x$
$d = h(X, pid_A, pid_B)$
$\overline{X} = AX^d, PS = B^{a+xd}$
$(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$
$H, \overline{X}, C_{AE} \leftarrow \mathsf{Enc}_{K_1}(H, pid_A||X||M))$
$PS = \overline{X}^b$
$(K_1, K_2) = KDF(PS, \overline{X}||pid_B)$
$\mathsf{Dec}_{K_1}(H, C_{AE}) = (pid_A, X, M)$
$d = h(X, pid_A, pid_B)$
Accept if pid_A is valid and $\overline{X} = AX^d$

Figure 1: Protocol structure of higncryption

$PS = B^{\rho(a+xd)} = \overline{X}^{\rho b}$ and the receiver will abort if $\overline{X} \notin G'$ or $PS = 1_G$. We note that the subgroup test can be waived, if the EXO queries are disallowed in the security model.

One-pass identity-concealed AKE. It is shown in [17, 16] 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 on the key generated for signcryption. When casting our higncryption scheme into one-pass identity-concealed AKE, 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. Another variant is to set the session-key of one-pass AKE to be $KDF(CDH(PS, X||pid_A||pid_B)$. This provides extra security guarantee, as X is exchanged in the encrypted form.

Flexible implementations. 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 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 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. *Note that the security of* light-*higncryption, even if sender's static secret-key is exposed to adversary, is not reduced to solving a quarter DL-problem. Actually, computing X from $X' = X^{h(X)}$ seems already to be hard, let along computing the exponent $x \leftarrow \{0,1\}^{\lceil |q|/4 \rceil} \cap Z_q^*$.*

Brief comparison. A brief comparison, among l-higncryption (referring to *light*-higncryption), (*full*) higncryption, (Zheng's) signcryption, and HOMQV, is in Table 2. For efficiency comparison, we only count the number of modular exponentiations (denoted "exp.") with standard computation technique. We note that, by using the speed-up technique of simultaneous exponentiation that may be less common, receiver efficiency of HOMQV and Zheng's signcryption is about 1.2 exo. "x-security" refers to that the leakage of DH-exponent should not expose session-key or sender's identity.

		l-higncryption	higncryption	signcryption	HOMQV
efficiency	sender	1.75exp.	2.5exp.	1exp.	2exp.
	receiver	1.5exp.	1.5exp.	2exp.	1.5exp.
forward ID-privacy		✓	✓	X	X
x-security		✓	✓	X	✓
receiver-deniability		✓	✓	X	✓

Table 2: Comparison with Zheng's signcryption and HOMQV

5. SECURITY PROOF OF HIGNCRYPTION

In the following security analysis, KDF and h are modeled to be random oracles (RO). 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 5.1. *The higncryption scheme presented in Figure 1 satisfies outsider unforgeability and insider confidentiality in the random oracle model, under the AEAD security and the GDH assumption (respectively, under the AEAD security and the HGDH and FGDL assumptions, for its flexible implementations when $x \leftarrow \mathcal{X}$ where $\mathcal{X}$ is a well-spread distribution over Z_q^*).*

5.1 Security Proof of Outsider Unforgeability

Suppose that, after a series of adaptive oracle queries to HO, UHO, EXO and Corrupt, with non-negligible probability the forger $\mathcal{A}^{OU}$ outputs a pair (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 $\frac{1}{n^2}$. 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 successful forgery before outputting (pid_{r^*}, C^*). All these assumptions are only for presentation simplicity.

Denote by pk_{s^*} (resp., pk_{r^*}) the public-key of the sender (resp., receiver) included in pid_{s^*} (resp., pid_{r^*}). Given (pid_{s^*}, pid_{r^*}), we construct a simulator S whose goal is to compute $CDH(pk_{s^*}, pk_{r^*})$ with the aid of a DDH-oracle, conditioned on that unforgeability is broken with non-negligible probability. 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 $pid_{s^*} \neq pid_{r^*}$.

For presentation simplicity, denote $pid_{s^*} = pid_{r^*} = pid_A$ with public-key $A = g^a$, where $a \leftarrow Z_q^*$ that is unknown to S. The simulator S takes $(params, pid_A)$ as input, and its goal is to compute $CDH(A, A) = g^{a^2}$ with the aid of a DDH oracle. It is well known that computing $CDH(A, A)$ is as hard as breaking the standard CDH assumption. 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 in order to deal with oracle queries made by the forger $\mathcal{A}^{OU}$ against pid_A.

First note S can perfectly handle all the Corrupt queries allowed in the security game, where neither $\mathsf{Corrupt}(pid_{s^*})$ nor $\mathsf{Corrupt}(pid_{r^*})$ is allowed.

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 $pid_s = pid_A$ below, 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 $\mathcal{L}_{DDH}$ that is maintained by S itself and is initiated to be empty. Note that in the latter 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 $KDF(PS', \overline{X}||pid_r)$, S checks, based on the list $\mathcal{L}_{DDH}$, whether $PS' = CDH(\overline{X}, pk_r)$ with the aid of the DDH oracle; if yes, it returns the pre-set value K_1.

In any case, the DH-exponent $x \leftarrow Z_q^*$ stored into $\mathcal{ST}_C$ 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, 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 other 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 "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 that the "failure" event does not occur, the simulation for UHO is perfect. Below, we show that the "failure" event can occur with at most negligible probability.

Note that the 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 K_1 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 of each other. By the security of AEAD as discussed in Section 2, $\mathsf{UHO}(pid_A, C)$ outputs $\perp$ with overwhelming probability. Thus, when the "failure" event occurs w.r.t. $\mathsf{UHO}(pid_A, C = (H, \overline{X}, C_{AE}))$ where pid_A is the receiver, with overwhelming probability 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}) makes up a valid AEAD ciphertext w.r.t. $K_1 = KDF(CDH(\overline{X}, A), \overline{X}||pid_A)$.

Below, we further consider two cases.

Case-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 the 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 the query $\mathsf{HO}(pid_A, pid_A, H', M')$. 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 S in simulating 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.

Case-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, "failure" occurs with negligible probability.

Then, we conclude that the failure event can occur with at most negligible probability, and consequently the view of $\mathcal{A}^{OU}$ in the simulation is indistinguishable from that in its real attack experiment. 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 $\mathsf{HO}(pid_{s^*}, pid_{r^*}, H^*, M^*)$. Here, pid_{s^*} and pid_{r^*} are the uncorrupted honest users, which are assumed to have been correctly guessed by the simulator for presentation simplicity. Recall that we are considering (the hardest case of) $pid_{s^*} = pid_{r^*} = pid_A$.

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 $\mathsf{unhigncrypt}$), 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 failure occurs with negligible probability in the UHO simulation, by the AEAD security the adversary $\mathcal{A}^{OU}$ 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 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 a new random output $d^{*\prime}$. Then, by the general forking lemma [3], 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 rewound run, where $\overline{X}^{*\prime} = AX^{*d^{*\prime}}$, and will make the query $KDF(CDH(\overline{X}^{*'}, pk_{r^*}), \overline{X}^{*'}||pid_{r^*}) = KDF(CDH(\overline{X}^{*'}, A), \overline{X}^{*'}||pid_A)$. 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 GDH assumption.

Unfortunately, a subtlety for correctly applying the forking lemma [3] 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 $(H, \overline{X}^*, C_{AE}) =\mathsf{higncrypt}(pid_i, pid_{r^*}, H, M)$, 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 $\mathsf{higncrypt}(pid_i, pid_{r^*}, H, M)$ 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, the collision event can occur with at most negligible probability assuming h is an RO. Specifically, for any pair of $(pid_i, pid_j, X) \neq (pid_{i'}, pid_{j'}, X')$, we have $\Pr[pk_i X^{h(X, pid_i, pid_j)} = pk_{i'} X'^{h(X', pid_{i'}, pid_{j'})}] \leq 2^{-l}$, 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 at most $C_t^2 \cdot 2^{-l} < t^2 \cdot 2^{-(l+1)}$ that is negligible. This saves the applicability of the forking lemma to higncryption, and then finishes the proof of outsider unforgeability.

5.2 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 given to $\mathcal{C}$ 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 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 a pair of equal-length messages (M_0, M_1), the associated data H and two pairs of public identity information $(pid^*_{s_0}, pid_{r^*})$ and $(pid_{s^*_1}, pid_{r^*})$ of equal lengths, 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 $\sigma \leftarrow \{0,1\}$ 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 $\overline{X}^* = AX^{*d^*}$. Then, $\mathcal{C}$ 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 based upon the list $\mathcal{L}_{DDH}$, whenever $\mathcal{C}$ finds that $\mathcal{A}^{IC}$ makes the query of $KDF(CDH(\overline{X}^*, B), \overline{X}^*||pid_B)$ 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 $G \setminus 1_G$ and perfectly hides the sender's identity information, even if $\mathcal{A}^{IC}$ knows $sk_{s^*_0}$ and $sk_{s^*_1}$ by user corruptions. Also, this ensures that $\mathcal{C}$ outputs "$\mathsf{failure}$" with negligible probability. Then, by the AEAD security, to win the insider confidentiality game, $\mathcal{A}^{IC}$ has to make the RO-query $KDF(CDH(\overline{X}^*, B), \overline{X}^*||pid_B)$ with non-negligible probability. *We remark that, for the proof here, it actually suffices if the distribution of $\overline{X}^*$ is only computationally indistinguishable from uniform distribution over $G \setminus 1_G$.* For the 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 presented in Appendix A, X^{*d^*} (and consequently $\overline{X}^*$) has distribution computationally indistinguishable from the uniform distribution over $G \setminus 1_G$ in the random oracle model.

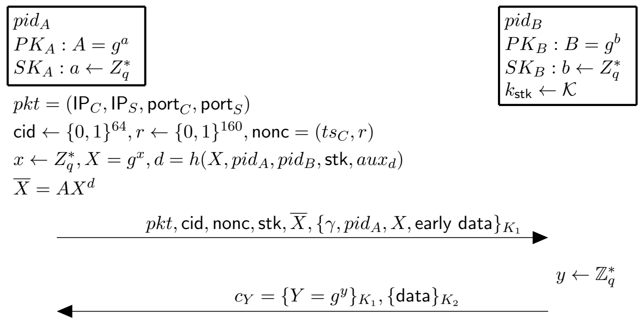
Then, $\mathcal{C}$ rewinds $\mathcal{A}^{IC}$ to the point of making the oracle query $h(X^*, pid_A, pid_B)$, redefines $d^{*\prime} = h(X^*, pid_A, pid_B)$, and re-runs $\mathcal{A}^{IC}$ from this rewinding point. By the forking lemma, with also non-negligible probability, $\mathcal{A}^{IC}$ will make the oracle query $KDF(CDH(\overline{X}^{*\prime}, B), \overline{X}^{*\prime}||pid_B)$ in the rewound run, where $\overline{X}^{*\prime} = AX^{*d^{*\prime}}$. From $\alpha = CDH(\overline{X}^*, B) = A^bX^{*bd^*}$ and $\beta = CDH(\overline{X}^{*\prime}, B) = A^bX^{*bd^{*\prime}}$, $\mathcal{C}$ computes $CDH(X^*, B) = (\alpha/\beta)^{(d^*-d^{*\prime})^{-1}}$, which violates the GDH assumption.

6. APPLICATIONS TO GOOGLE'S 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 [24]. Here, for presentation simplicity, the following protocol description does not fully coincide with (and, actually, omits many of) the technical details of QUIC.

QUIC supports two connection modes [24]: 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; And 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 IP addresses, port numbers, $(\mathsf{IP}_C, \mathsf{IP}_S, \mathsf{port}_C, \mathsf{port}_S)$ and the time-stamp ts_S of the server. stk plays a role similar to that of the session ticket in TLS, which can be used by the client in later 0-RTT connections (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 Figure 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. "$\{\cdots\}_K$" 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.



pid_A
$PK_A : A = g^a$
$SK_A : a \leftarrow Z^*_q$
pid_B
$PK_B : B = g^b$
$SK_B : b \leftarrow Z^*_q$
$k_{\mathsf{stk}} \leftarrow \mathcal{K}$
$pkt = (\mathsf{IP}_C, \mathsf{IP}_S, \mathsf{port}_C, \mathsf{port}_S)$
$\mathsf{cid} \leftarrow \{0,1\}^{64}, r \leftarrow \{0,1\}^{160}, \mathsf{nonc} = (ts_C, r)$
$x \leftarrow Z^*_q, X = g^x, d = h(X, pid_A, pid_B, \mathsf{stk}, aux_d)$
$\overline{X} = AX^d$
$pkt, \mathsf{cid}, \mathsf{nonc}, \mathsf{stk}, \overline{X}, \{\gamma, pid_A, X, \text{early data}\}_{K_1}$
$y \leftarrow \mathbb{Z}^*_q$
$c_Y = \{Y = g^y\}_{K_1}, \{\mathsf{data}\}_{K_2}$

Figure 2: 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. The 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, in particular ts_C, protected by the AEAD encryption, rather than being sent in clear.

7. SECURITY DEFINITIONAL FRAMEWORK FOR IDENTITY-CONCEALED AKE

Let n denote the largest number of users in the system, and $\mathcal{U} = \{U_1, \cdots, U_n\} = \mathsf{HONEST} \bigcup \mathsf{DISHONEST}$ be 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). 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. The certificate $cert_i$, for each user of identity $id_i \in \mathcal{U}$ and public-key pk_i, $1 \leq i \leq n$, is issued by a single certificate authority (CA). The public identity information for each user is set to be $pid_i = (id_i, pk_i, cert_i)$. Again, we assume the public identity information for all the users to be of equal length (otherwise, we need to employ length-hiding AEAD as underlying building tool in the protocol construction).

We assume each session has a unique session-identifier sid that is simply assigned by an incremental counter. Setting session identifiers via a counter is just an artefact for security modeling. 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 them are originally initialized to be the empty string (meaning "undefined"), and are assigned during session run according to the protocol specifications.

A session held at user U_i is called 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 during the 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.

7.1 Adversarial Setting

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

$\mathsf{Initiator}$: This oracle keeps a counter CTR_I that is initiated to be 0, and keeps in private a random bit $\sigma \leftarrow \{0,1\}$.

- Upon receiving a special "(Start, U_i)" instruction, $1 \leq i \leq n$, for an *honest yet uncorrupted* user $U_i \in \mathsf{HONEST} \setminus \mathsf{CORRUPTED}$ (otherwise, this instruction is ignored),[3] it sets $CTR_I := CRT_I + 1$, creates a session for the user U_i with session-identifier $sid_I = CTR_I$, and returns $(sid_I, msg^{(1)}_{sid_I})$, where $msg^{(1)}_{sid_I}$ denotes the first protocol message of sid_I. It also creates, and keeps in private, the local states $peer_{sid_I}$, ST_{sid_I} and SK_{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 with the next protocol message if msg^*_I is not the last protocol message for that session; otherwise, it works according to the protocol specifications.

- Upon receiving an instruction of "$(sid_I, \mathsf{ST\text{-}Exposure})$", it returns the value stored in ST_{sid_I}, if sid_I is incomplete; 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.

- Upon receiving "$(sid_I, \mathsf{SK\text{-}exposure})$", it returns 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 $sklen$ denotes the length of session-key.

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

- When receiving an instruction "$(\mathsf{Start}, U_j, msg^{(1)}_R)$", $1 \leq j \leq n$, for an honest yet uncorrupted user U_j (otherwise, the instruction is ignored), it sets $CTR_R := CRT_R{+}1$, creates a session for the user U_j with session-identifier $sid_R = CTR_R$; it then treats $msg^{(1)}_R$ as the first-round incoming message of session sid_R and returns $(sid_R, msg^{(2)}_R)$, where $msg^{(2)}_R$ denotes the second-round message of sid_R. It also creates, and keeps in private, the local states $peer_{sid_R}$, ST_{sid_R} and SK_{sid_R}. We remark that the same user can be indicated to be both initiator (in one session held at $\mathsf{Initiator}$) and responder (in another session held at $\mathsf{Responder}$).

- Upon receiving an instruction "(sid_R, msg^*_R)", it treats msg^*_R as the incoming message for session sid_R, and promptly responds with the next protocol message if msg^*_R is not the last protocol message of that session.

- Upon receiving "$(sid_R, \mathsf{ST\text{-}Exposure})$", it returns the value stored in ST_{sid_R}, if session sid_R is incomplete; otherwise, it ignores this instruction.

- Upon receiving "$(\mathsf{Test}, U_{t_0}, U_{t_1}, msg^{(1)}_R)$" 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 "$(\mathsf{Start}, U_t, msg^{(1)}_R)$" instruction, where the session-identifier set is denoted as sid_T for presentation simplicity. We remark that the Test-type query can be made by the adversary for only one time during its attack, exclusively against $\mathsf{Initiator}$ or $\mathsf{Responder}$.

- Upon receiving "$(sid_R, \mathsf{SK\text{-}exposure})$", it returns 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 the value stored in $peer_{sid}$, if sid is an existing session held at $\mathsf{Initiator}$ or $\mathsf{Resonder}$ (otherwise, the query is ignored). Note that the value returned may be an empty string, in case the peer of that session has not been determined.

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

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

Corrupt: Upon receiving "U_i", $1 \leq i \leq n$, if $U_i \in$ HONEST it returns 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 CORRUPTED = CORRUPTED $\bigcup\{U_i\}$. If $U_i \notin$ HONEST, the query is ignored.

Register: Upon receiving "(U_i, honest), where U_i is a new user not in HONEST $\bigcup$ DISHONEST, it generates the public/secret key pair and gets the certificate for U_i (with the aid of CA), returns pid_i, and sets HONEST = HONEST $\bigcup\{U_i\}$. Upon receiving "$((U_j, pk_j), \mathsf{dishonest})$ for a new user U_j, it gets the certificate for (U_j, pk_j) (with the aid of CA), returns pid_j, and sets DISHONEST = 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 the end of the attack, $\mathcal{A}$ outputs a bit σ'.

7.2 CAKE Security Definition

In our model, the label of a session is specified to be part of the session transcript. 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 $peer_{sid_T} = U_k \in$ HONEST, $1 \leq k \leq n$, where U_t *and* U_k *may be the same user*. 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:

- U_{t_0} or U_{t_1} is corrupted via the Corrupt query,[4] or ST_{sid_T} was exposed via the $(ST_{sid_T}, \mathsf{ST\text{-}exposure})$ query;[5]
- The static secret-key of U_k is exposed;[6]
- The query $(sid, \mathsf{SK\text{-}exposure})$, where $sid \in \{sid_T, sid'_T\}$ (in case the matching session exists), was issued;
- The query $\mathsf{Peer}(sid'_T)$ was issued.

Definition 7.1 (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. (1) There exist more than two sessions of the same session label. (2) There exist two matching sessions: session sid held at user U_i and session sid$'$ held at user U_j (where U_i may be equal to 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-view mismatching:** *either $peer_{sid} \neq \perp \wedge peer_{sid} \neq U_j$, or $peer_{sid'} \neq \perp \wedge peer_{sid'} \neq U_i$.*[7]

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

ID-concealed session-key (ICSK) security: *On condition that the test-session sid_T is completed and unexposed, both of the following quantities are negligible:*

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

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

Discussion. Note that label-security implies the security against unknown key share (UKS) attack. As the static secret-keys of both U_{t_0} and U_{t_1} can 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 views and the same session-key, while unmatched sessions must have different (computationally 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 cast 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 query to Peer. This way of formulation, on the one hand, allows a more powerful adversary, and incorporates the post-ID CK-framework [9] as a special case on the other hand.

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.

7.3 Adaption to Unilateral CAKE (UCAKE)

For simplicity, we assume only the responder (server) authenticates it to the initiator (client), while the client may not necessarily possess public identity information. 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

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

[5]If $U_{t_\sigma} = U_k$, ST_{sid_T} is allowed to be exposed.

[6]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.

[7]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 them 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.

$\mathsf{Corrupt}(U_i)$ 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 client 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 $\mathsf{Responder}$ they must both have public-keys.[8] For presentation consistency with the definitional framework of CAKE, we still allow the adversary to indicate the session holder when starting a session at $\mathsf{Initiator}$.

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 $\mathsf{Initiator}$) or $peer_{sid_T}$ is an empty string representing undefined (if sid_T is run at $\mathsf{Responder}$). 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 $\mathsf{Corrupt}$ query,[9] or ST_{sid_T} was exposed via the $(ST_{sid_T}, \mathsf{ST\text{-}exposure})$ query.[10]
- The query $(sid, \mathsf{SK\text{-}exposure})$, $sid = sid_T$ or $sid = sid'_T$, was issued.
- sid_T is run at $\mathsf{Initiator}$ but the static secret-key of U_k is exposed.[11]
- sid_T is run at $\mathsf{Responder}$ 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.

Definition 7.2 (UCAKE-security). *The definition of strong UCAKE-security is almost identical to that of strong CAKE-security, with the following modifications:*

Peer-view mismatching: $peer_{sid} \neq \perp \bigwedge peer_{sid} \neq U_j$ if U_j is the responder*, or* $peer_{sid'} \neq \perp \bigwedge peer_{sid'} \neq U_i$ if U_i is the responder.

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

8. CONSTRUCTION AND ANALYSIS OF UCAKE AND CAKE

Our higncryption can be straightforwardly adapted into a two-round UCAKE protocol (or *identity-concealed* SACCE protocol [21]), by replacing the public identity information of the receiver with a randomly generated DH-component. The higncryption based UCAKE protocol is briefly described in Figure 3, where K_2 serves as the resultant session-key, [12] H_B denotes the associated data from the server for AEAD.

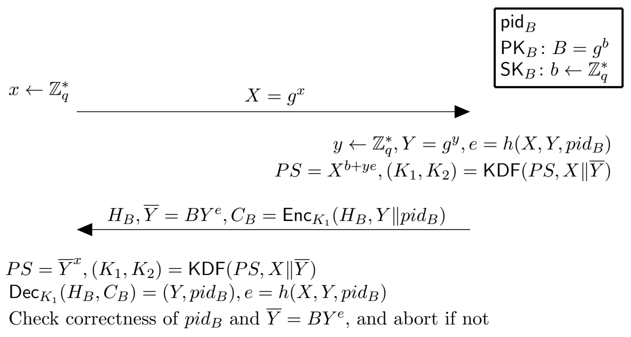
[8]Actually, if the test-query is against $\mathsf{Initiator}$, specifying (U_{t_0}, U_{t_1}) does not make sense for UCAKE. The treatment is for presentation consistency with the model of CAKE.

[9]Exposing the static secret-keys of both U_{t_0} and U_{t_1} does not expose the test-session.

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

[11]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.

[12]For identity-concealed SACCE, K_1 serves the session-key while K_2 is set to be empty.



pid_B
$\mathsf{PK}_B: B = g^b$
$\mathsf{SK}_B: b \leftarrow \mathbb{Z}_q^*$
$x \leftarrow \mathbb{Z}_q^*$
$X = g^x$
$y \leftarrow \mathbb{Z}_q^*, Y = g^y, e = h(X, Y, pid_B)$
$PS = X^{b+ye}, (K_1, K_2) = \mathsf{KDF}(PS, X \| \overline{Y})$
$H_B, \overline{Y} = BY^e, C_B = \mathsf{Enc}_{K_1}(H_B, Y \| pid_B)$
$PS = \overline{Y}^x, (K_1, K_2) = \mathsf{KDF}(PS, X \| \overline{Y})$
$\mathsf{Dec}_{K_1}(H_B, C_B) = (Y, pid_B), e = h(X, Y, pid_B)$
Check correctness of pid_B and $\overline{Y} = BY^e$, and abort if not

Figure 3: Protocol structure of higncryption-based UCAKE

We assume the server always performs subgroup membership test for the incoming DH-exponent X explicitly or implicitly. In practice, we recommend the following protocol variant with implicit subgroup test, where $PS = X^{\rho(b+ye)}$ and the server will abort if $X \notin G'$ or $PS = 1_G$. We also note that the subgroup test can be waived, if oracle query to EXO is denied in the UCAKE-security definition.

In comparison with QUIC, (EC-DSA based) TLS1.3 and OPTLS, our higncryption-based UCAKE protocol is signatureless, and more efficient;[13] Moreover, it has forward ID-privacy, receiver deniability, and strong resilience to exposure of intermediate state (whether y or $b + ye$, but not both of them); Finally, it enjoys flexible implementations and deployments.

Theorem 8.1. *The protocol presented in Fig. 3 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. 3, its label is defined to be $X \| \overline{Y}$. The local state ST_{sid} is specified to be (Y, y) (if sid is run at $\mathsf{Responder}$) or x (if sid is run at $\mathsf{Initiator}$), which can be offline computed.

Lemma 8.1. *Assuming $h : \{0,1\}^* \to \{0,1\}^l \cap Z_q^*$ is 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 $\{aux' \in \{0,1\}^*, Y' \in G, pid_k = (id_k, pk_k, cert_k)\}$, such that $pk_j Y^{h(aux,Y,pid_j)} = pk_k Y'^{h(aux',Y',pid_k)}$ but $\{aux, Y, pid_j\} \neq \{aux', Y', pid_k\}$.*

Proof (of Lemma 8.1). For any pair of different $\{aux, Y, pid_j\}$ and $\{aux', Y', pid_k\}$, $1 \leq j, k \leq n$, 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 is an RO. Then, for any PPT algorithm who makes at most m oracle queries to h, it succeeds with probability at most $\frac{m^2}{2^l-1}$ that is negligible. $\square$

We first prove the label-security. It is straightforward that, with overwhelming probability, not more than two sessions can have the same label. Let session sid held at user U_i and session sid' held at user U_j, $1 \leq i, j \leq n$, be matching (i.e., they have the same session label $X \| \overline{Y}$). Note that, for honestly generated X and $\overline{Y}$, $X \neq \overline{Y}$ with overwhelming probability, which holds even if y is taken from a well-spread distribution over Z_q^* with min-entropy greater than $\omega(\log |q|)$. Then, label matching implies that U_i and U_j cannot play the same session role, and the two matching sessions must

[13]For UCAKE based on *medium* (resp., *light*) higncryption, the server performs only 2 (resp., 1.75) exponentiations.

have the same session-key. Finally, the property of peer-view matching is established with Lemma 8.1.

Next, we prove the ID-concealed session-key (ICSK) security. Let the test-session sid_T, held at the uncorrupted client user $U_t = U_{t_\sigma}$ for $\sigma \leftarrow \{0,1\}$, be completed and unexposed, with peer server responder $peer_{sid_T} = U_k \in$ HONEST. Breaking *impersonation* security (i.e., making the *exposed* test-session have no matching session) implies that a PPT adversary can impersonate 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 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. This case is ruled out by Lemma 8.1.

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

Proof (of Lemma 8.2). Assuming a PPT adversary $\mathcal{A}$ who makes Case-2 occur with non-negligible probability, we construct a GDH-solver $\mathcal{S}$. $\mathcal{S}$ 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. For presentation simplicity, $\mathcal{S}$ randomly guesses the victim responder user U_k, sets its public-key to be B and sets pid_B accordingly for U_k. $\mathcal{S}$ 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 (related to U_k of pid_B) as follows.

When $\mathcal{A}$ starts a session via the oracle query "$(\mathsf{Start}, U_k, X' \in G)$", $\mathcal{S}$ works as U_k does, except that (K_1, K_2) are set to be random strings chosen by $\mathcal{S}$ itself. $\mathcal{S}$ uses its DDH-oracle to ensure the consistency of the random oracle KDF.

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

It is easy to check that the view of $\mathcal{A}$ under the run of $\mathcal{S}$ is identical to that in $\mathcal{A}$'s real attack, which means Case-2 occurs also with non-negligible probability 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 sid_T 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 [3], Case-2 occurs in the rewound run also with non-negligible probability, from which $\mathcal{S}$ will get $CDH(X, \overline{Y}') = X^{b+y'e'}$. 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. $\square$

Finally, we prove the ID-SK indistinguishability. For UCAKE protocol, we only need to consider ID-indistinguishability when sid_T is run at the target server user U_t. As the test-session sid_T is completed and unexposed, and has matching session, from now on denote by $X||\overline{Y}$ the session label of sid_T, where $\overline{Y} = U_t Y^{h(X,Y,U_t)}$ and $Y = g^y$ are generated by U_t. Note that $ST_{sid_T} = (Y, y)$ is unexposed. In this case, the first observation is that $\overline{Y}$ perfectly hides the responder's identity assuming h is an RO. Furthermore, it is easy to see 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 RO model.

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

The first case is that sid_T is run at $\mathsf{Responder}$. In this case, as the matching session sid'_T must exist and neither ST_{sid_T} nor $ST_{sid'_T}$ is exposed, the ability to break ID-SK indistinguishability implies the ability to break the GDH assumption assuming KDF is an RO. Specifically, we can construct a GDH-solver $\mathcal{S}$ which, on input (X, Y) (w.r.t. the session label $X||\overline{Y}$), can with non-negligible probability compute $CDH(X, Y)$ with the aid of a DDH-oracle.

The second case is that sid_T is run at $\mathsf{Initiator}$. By the impersonation security, the matching session sid' must exist at the server user U_k. In this case, the ability of breaking ID-SK indistinguishability is also reduced to the ability of breaking the GDH assumption. Denote by B the public-key of U_k for presentation simplicity. The GDH-solver $\mathcal{S}$ takes (X, B) as input, and can with non-negligible probability compute $CDH(X, B)$ with the aid of a DDH-oracle. This finishes the proof of Theorem 8.1. $\square$

COROLLARY 8.1. *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. 3, 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 Appendix A), in the random oracle model.*

Finally, the (parallel) composition of two UCAKE renders us a three-round protocol, which is strongly CAKE-secure under the AEAD security and the GDH assumption in the random oracle model. The details are deferred to the full version due to space limitation.

9. ACKNOWLEDGEMENT

We thank the anonymous referees for their insightful comments. We are grateful to Boru Gong, Yue Guo and Hongbing Wang for many helpful discussions.

10. 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 G. Neven. Multi-Signatures in the Plain Ppublic-Key Model and a General Forking Lemma. *ACM CCS*, 2006: 390-399.

[4] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *CRYPTO* 1993: 273-289.

[5] C. Brzuska, M. Fischlin, B. Warinschi, and S.C. Williams. Composability of Bellare-Rogaway key exchange protocols. *ACM CCS* 2011: 51-62.

[6] C. Brzuska, N.P. Smart, B. Warinschi, G.J. Watson. An Analysis of the EMV Channel Establishment Protocol. *ACM CCS* 2013: 373-386.

[7] R. Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. *CRYPTO* 1997: 455-469.

[8] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. *EUROCRYPT* 2001: 453-474.

[9] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. *CRYPTO* 2002: 143-161.

[10] A.W. Dent. Hybrid Cryptography. Cryptology ePrint Archive, Report No. 2004/210.

[11] Y. Dodis and J.H. An. Concealment and Its Applications to Authenticated Encryption. *EUROCRYPT* 2003: 312-329.

[12] 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.

[13] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, Draft-12, 2016. https://tools.ietf.org/html/draft-ietf-tls-tls13-12

[14] J. Fan, Y. Zheng, and X. Tang. A Single Key Pair is Adequate for the Zheng Signcryption. *ACISP* 2011: 371-388.

[15] M.Fischlin and F. Günther. Multi-Stage Key Exchange and the Case of Googleąŕs QUIC Protocol. *ACM CCS* 2014: 1193-1204.

[16] M. Gorantla, C. Boyd, J. Gonzalez Nieto. On the Connection Between Signcryption and One-Pass Key Establishment. *Cryptography and Coding* 2007: 277-301.

[17] S. Halevi and H. Krawczyk. One-Pass HMQV and Asymmetric Key-Wrapping. *PKC* 2011: 317-334.

[18] T. Jager and J. Schwenk. On the Equivalence of Generic Group Models. *ProvSec* 2008: 200-209.

[19] H. Krawczyk. SIGMA: The "Sign-and-Mac" Approach to Authenticated Diffie-Hellman and Its Use in the IKE-protocols. *CRYPTO* 2003: 400-425.

[20] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. *CRYPTO* 2005: 546-566.

[21] H. Krawczyk, K.G. Paterson and H. Wee. On the Security of the TLS Protocol: A Systematic Analysis. *CRYPTO* 2013: 429-448.

[22] Hugo Krawczyk and Hoeteck Wee. The OPTLS Protocol and TLS 1.3. *EuroS&P* 2016: 81-96.

[23] B. Libert and J.-J. Quisquater. Efficient Signcryption with Key Privacy from Gap DiffieÍCHellman Groups. *PKC* 2004: 187-200.

[24] R. Lychev, S. Jeroy, A. Boldyrevaz and C. Nita-Rotarux. How Secure and Quick is QUIC? Provable Security and Performance Analyses. *IEEE S&P* 2015: 214-231.

[25] U. Maurer. Abstract Models of Computation in Cryptography. *Cryptography and Coding* 2005: 1-12.

[26] U. Maurer and S. Wolf. Lower Bounds on Generic Algorithms in Groups. *EUROCRYPT* 1998: 72-84.

[27] A. Menezes, M. Qu, and S. Vanstone. Some New Key Agreement Protocols Providing Mutual Implicit Authentication. *SAC* 1995: 70–88.

[28] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. *ASIACRYPT* 2011: 372-389.

[29] P. Rogaway. Authenticated-Encryption with Associated-Data. *ACM CCS* 2002: 98-107.

[30] J. Roskind. Quick UDP Internet Connections: Multiplexed Stream Transport over UDP. 2012.

[31] C. P. Schnorr. Small Generic Hardcore Subsets for the Discrete Logarithm. *Information processing Letters* 79(2): 93-98, 2001.

[32] J. T. Schwartz. Fast Probabilistic Algorithms for Verifications of Polynomial Identities. *Journal of the ACM*, 27(3): 701-717, 1980.

[33] C.-H. Tan. Analysis of Improved Signcryption Scheme with Key Privacy. *Information Processing Letters* 99(4): 135-138, 2006.

[34] G. Yang, D.S. Wong and X. Deng. Analysis and Improvement of a Signcryption Scheme with Key Privacy. *ISC* 2005: 218-232.

[35] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. *EUROCRYPT* 1997: 256-266.

[36] A. C. Yao and Y. Zhao. OAKE: A New Family of Implicitly Authenticated Diffie-Hellman Protocols. *ACM CCS* 2013: 1113-1128.

[37] Y. Zheng. Digital signcryption or how to achieve cost(Signature & encryption) $\ll$ cost(Signature) +cost(Encryption). *CRYPTO* 1997: 165-179.

APPENDIX

A. FLEXIBLE DISCRETE LOGARITHM (FDL) AND RELATED PROBLEMS

We say a distribution $\mathcal{D}$ is *well-spread* over a subset $\mathcal{S} \subseteq Z_q^*$, if the min-entropy $\lambda_{\mathcal{D}} = -\log(\max_{s\in\mathcal{S}}(\Pr[\mathcal{D} = s])) > \omega(\log|q|)$ when $|q|$ is sufficiently large. Let $\mathcal{X}$ and $\mathcal{Y}$ be well-spread distributions over some subsets of Z_q^* with min-entropy $\lambda_{\mathcal{X}}$ and $\lambda_{\mathcal{Y}}$ respectively. 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 $CDH(X = g^x, Y = g^y)$ 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), where $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$ and $z \leftarrow Z_q^*$. The flexible gap Diffie-Hellman (FGDH) problem is to compute $CDH(X = g^x, Y = g^y)$, where $x \leftarrow \mathcal{X}$ 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, where $x \leftarrow \mathcal{X}$, $y \leftarrow \mathcal{Y}$ and $z \leftarrow Z_q^*$, with the aid of a DDH oracle. Notice the difference between FGDL and FDDH, where for FGDL g^x and g^y are not given to the distinguisher as input.

Clearly, the traditional problems of DL, CDH, DDH, GDH are special cases of their flexible counterparts, when $\mathcal{X}$ and $\mathcal{Y}$ are constrained to be the uniform distribution over Z_q^* with min-entropy $\log(q-1)$. The work [7] 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^* but $\mathcal{Y}$ is the uniform distribution over Z_q^*; HDDH 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^* but $\mathcal{Y}$ is the uniform distribution over Z_q^*. Actually, as shall see, what we need in this work, *only for the provable security of flexible efficient implementations*, are the HGDH assumption and the FGDL assumption, where FGDL assumption is used only for proving forward ID-privacy of the flexible implementations.

We first prove the following lemma, which is a generalized version of the Schwartz-Shoup lemma [32, 35].

LEMMA A.1. *Let $\mathcal{X}_i$, $1 \le i \le k$, be* well-spread *distribution over subset $\mathcal{S}_i \subseteq Z_q^*$ with min-entropy $\lambda_i > \omega(\log |q|)$. 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 is taken independently from $\mathcal{X}_i$ (i.e., $x_i \leftarrow \mathcal{X}_i$) for $1 \le i \le k$, is at most $d \cdot (2^{-\lambda_1} + \cdots + 2^{-\lambda_k})$.*

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

Now, supposing this lemma holds for the case of $k-1$, we consider the case of k. Let $1 \le \vartheta \le d$ 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 inductive hypothesis, $\Pr[e_\vartheta(x_1, \cdots, x_{k-1}) = 0] \le (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}})$, 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 the values $(x_1, \cdots, x_k)$ satisfying P_k is non-zero), there are at most ϑ solutions of x_k such that $P_k(x_k) = 0$. As $\lambda_i > \omega(\log|q|)$, $1 \le i \le k$, and d and k are polynomial in $|q|$, we have that $d(2^{-\lambda_1} + \cdots + 2^{-\lambda_k}) < 1$ (when $|q|$ is sufficiently large). In summary, for $x_i \leftarrow \mathcal{X}_i$, $1 \le i \le k$, $\Pr[P(x_1, \cdots, x_k) = 0] \le (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}}) + (1 - (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}}))\vartheta 2^{-\lambda_k} \le (d-\vartheta)(2^{-\lambda_1} + \cdots + 2^{-\lambda_{k-1}}) + \vartheta 2^{-\lambda_k} < d(2^{-\lambda_1} + \cdots + 2^{-\lambda_k})$. □

For presentation simplicity, in the following analysis we use Maurer's generic group model [25] that is actually equivalent to Shoup's model [35, 18], 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 $\max\{2^{-\lambda_\mathcal{X}}, 2^{-\lambda_\mathcal{Y}}\}$ that is negligible.

THEOREM A.1. *For an algorithm of τ generic steps for solving the FGDH problem, its success probability is upper bounded by $\tau^3(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$ 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 $L[l]$ the value stored in the l-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, $1 \le i \le \tau$, 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 the previous GG-oracle accesses. The value $F_i(x, y)$ is not returned to $\mathcal{A}$ directly, but is stored into a position in the internal list L where the position index for storing $F_i(x, y)$ is, however, determined 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 the binary result depending on whether $L[i] = L[j]$ or not. For adversary against the FGDH problem, the adversary $\mathcal{A}$ is additionally allowed to query the GG-oracle with (i, j, k), where the GG-oracle, corresponding to the DDH oracle in this case, returns 1 if and only if $L[i]L[j] = L[k]$.

As discussed in [25], in this generic group model, we only need to consider non-adaptive adversaries; And there are only two approaches for $\mathcal{A}$ to win, other than simply guessing x or y that is disregarded in the analysis. One approach is to cause two different F_i and F_j to collide, in the sense that $a_i x + b_i y + c_i = a_j x + b_j y + c_j$ 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 A.1, 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 win is to cause, for some (i, j, k), the *non-zero* polynomial $F_i F_j - F_k = 0$ (note that, if the polynomial $F_i F_j - F_k$ is a zero-polynomial, it leaks nothing). That is, $(a_i x + b_i y + c_i)(a_j x + b_j y + c_j) - (a_k x + b_k y + c_i) = 0$ for $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$. Note that $F_i F_j - F_k$ is a quadratic polynomial. By Lemma A.1 with $d = 2$, this event occurs with probability at most $C_\tau^3(2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}}))$.

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

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

Proof. Let x (resp., y) be taken according to the well-spread distribution $\mathcal{X}$ (resp., $\mathcal{Y}$), and $z \leftarrow Z_q^*$. 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 L, where $T_\sigma = xy$ and $T_{1-\sigma} = z$ for a random bit $\sigma \leftarrow \{0, 1\}$. 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 σ, with the aid of a DDH oracle.

For the i-th GG-oracle access corresponding to a group operation, $1 \le i \le \tau$, 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 $a_i, b_i, c_i \in Z_q$ are determined by the 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 σ) 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 A.1, the advantage is at most $\frac{1}{2} + C_\tau^2(2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})) + C_\tau^3(4(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})) < \frac{1}{2} + \tau^3(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})$. $\square$

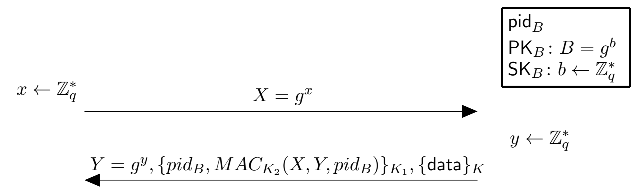
In the full version, we also prove the following theorem regarding the intractability of FDL, FDL and FDDH.

THEOREM A.3. *For an algorithm of τ generic steps, its success probability is upper bounded by $\tau^2 2^{-\lambda_\mathcal{X}}$ (for solving the FDL problem), $\tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}})$ (for solving the FCDH problem), and $\frac{1}{2} + \tau^2(2^{-\lambda_\mathcal{X}} + 2^{-\lambda_\mathcal{Y}} + \frac{1}{q-1})$ (for solving the FDDH problem), in the generic group model.*

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 [31], 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 roughly $\frac{\tau}{|H|}$ (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 the traditional DL defined over Z_q^* (at least for generic algorithms)! But the result [31] critically relies on uniform distribution over the subset H, while our result is for any distribution with super-logarithmic min-entropy.

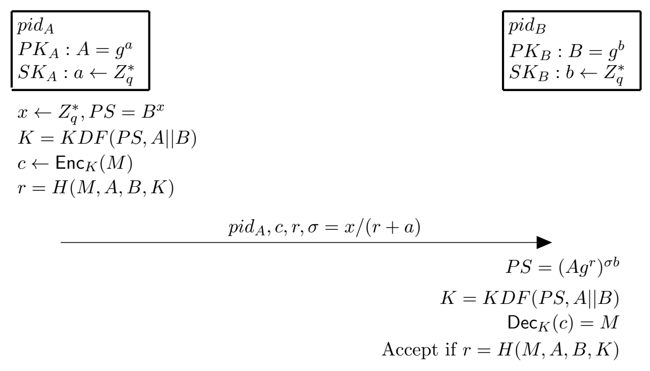
The result about FDL and related problems has two consequences. On the one hand, it indicates that cryptosystems based on the traditional DL and related problems have strong resilience to randomness leakage, in the sense they are still secure as long as super-logarithmic min-entropy remains with each exponent secrecy. On the other hand, it allows more flexible and efficient implementations of cryptosystems based on DL and related problems, according to priorities and tradeoffs among security and efficiency in different application scenarios.

B. RELATED PROTOCOLS



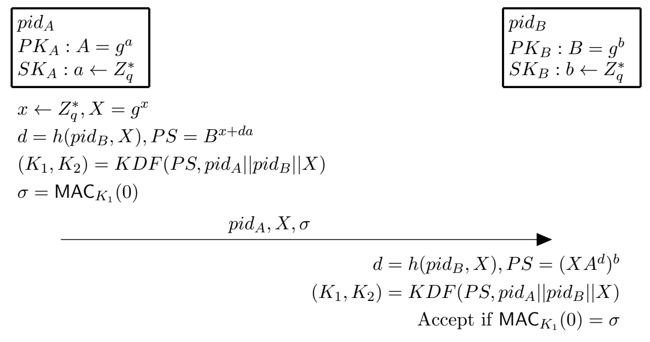
pid_B
PK_B: $B = g^b$
SK_B: $b \leftarrow \mathbb{Z}_q^*$
$x \leftarrow \mathbb{Z}_q^*$
$X = g^x$
$y \leftarrow \mathbb{Z}_q^*$
$Y = g^y, \{pid_B, MAC_{K_2}(X, Y, pid_B)\}_{K_1}, \{\mathsf{data}\}_K$

Figure 4: Basic structure of OPTLS, where K_1 is derived from g^{xy}, K_2 from g^{xb}, and K from both g^{xb} and g^{xy}.



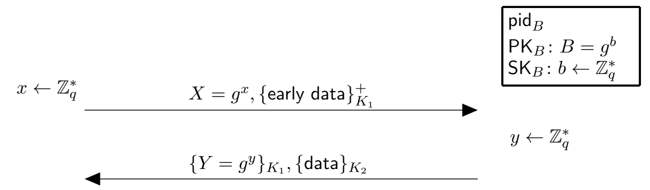
pid_A
$PK_A : A = g^a$
$SK_A : a \leftarrow Z_q^*$
pid_B
$PK_B : B = g^b$
$SK_B : b \leftarrow Z_q^*$
$x \leftarrow Z_q^*, PS = B^x$
$K = KDF(PS, A||B)$
$c \leftarrow \mathsf{Enc}_K(M)$
$r = H(M, A, B, K)$
$pid_A, c, r, \sigma = x/(r + a)$
$PS = (Ag^r)^{\sigma b}$
$K = KDF(PS, A||B)$
$\mathsf{Dec}_K(c) = M$
Accept if $r = H(M, A, B, K)$

Figure 5: Zheng's signcryption, where $H : \{0, 1\}^* \rightarrow Z_q$.



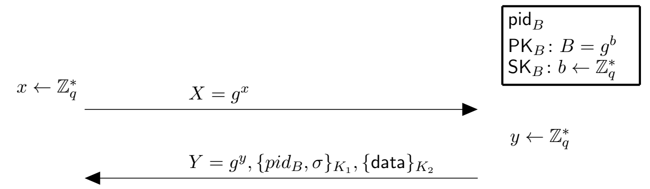
pid_A
$PK_A : A = g^a$
$SK_A : a \leftarrow Z_q^*$
pid_B
$PK_B : B = g^b$
$SK_B : b \leftarrow Z_q^*$
$x \leftarrow Z_q^*, X = g^x$
$d = h(pid_B, X), PS = B^{x+da}$
$(K_1, K_2) = KDF(PS, pid_A||pid_B||X)$
$\sigma = \mathsf{MAC}_{K_1}(0)$
pid_A, X, σ
$d = h(pid_B, X), PS = (XA^d)^b$
$(K_1, K_2) = KDF(PS, pid_A||pid_B||X)$
Accept if $\mathsf{MAC}_{K_1}(0) = \sigma$

Figure 6: HOMQV, with session-key set to be K_2.



pid_B
PK_B: $B = g^b$
SK_B: $b \leftarrow \mathbb{Z}_q^*$
$x \leftarrow \mathbb{Z}_q^*$
$X = g^x, \{\mathsf{early\ data}\}_{K_1}^+$
$y \leftarrow \mathbb{Z}_q^*$
$\{Y = g^y\}_{K_1}, \{\mathsf{data}\}_{K_2}$

Figure 7: Basic structure of QUIC. K_1 (resp., session-key K_2) is derived from $CDH(X, B)$ (resp., $CDH(X, Y)$) and some auxiliary input."$\{\ \}_K^+$" means it is optionally generated and sent only for 0-RTT mode.



pid_B
PK_B: $B = g^b$
SK_B: $b \leftarrow \mathbb{Z}_q^*$
$x \leftarrow \mathbb{Z}_q^*$
$X = g^x$
$y \leftarrow \mathbb{Z}_q^*$
$Y = g^y, \{pid_B, \sigma\}_{K_1}, \{\mathsf{data}\}_{K_2}$

Figure 8: Basic structure of TLS1.3. σ is server's signature on the hash of session transcript, K_1 and K_2 are derived from $CDH(X, Y)$ and some auxiliary information.