A New Family of Implicitly Authenticated Diffie-Hellman Protocols*

Andrew C. Yao [†] Yunlei Zhao[‡]

Abstract

Cryptography algorithm standards play a key role both to the practice of information security and to cryptography theory research. Among them, the MQV and HMQV protocols ((H)MQV, in short) are a family of *implicitly authenticated* Diffie-Hellman key-exchange (DHKE) protocols that are among the most efficient and are widely standardized. In this work, from some new perspectives and under some new design rationales, and also inspired by the security analysis of HMQV, we develop a new family of practical implicitly authenticated DHKE (IA-DHKE) protocols, which enjoy notable performance among security, efficiency, privacy, fairness and easy deployment. We make detailed comparisons between our new protocols and (H)MQV, showing that the newly developed protocols outperform HMQV in most aspects. Very briefly speaking, we achieve:

- The most efficient provably secure IA-DHKE protocol to date, and the first online-optimal provably secure IA-DHKE protocols.
- The first IA-DHKE protocol that is provably secure, *resilience to the leakage of DH components and exponents*, under merely standard assumptions without additionally relying on the knowledge-of-exponent assumption (KEA).
- The first provably secure privacy-preserving and computationally fair IA-DHKE protocol, with privacy-preserving properties of reasonable deniability and post-ID computability and the property of session-key computational fairness.

Guided by our new design rationales, in this work we also formalize and introduce some new concept, say session-key computational fairness (as a complement to session-key security), to the literature.

1 Introduction

Diffie-Hellman key-exchange (DHKE) protocols [23] marked the birth of modern cryptography, and are one of the main pillars of both theory and practice of cryptography [15]. Among them, the (H)MQV protocols [50, 45, 42, 51] are among the most efficient DHKE protocols that provide implicitly mutual authentications based upon public-key cryptography, and are widely standardized [3, 4, 37, 38, 56, 57, 63]. In particular, it has been announced by the US National Security Agency as the key exchange mechanism underlying "the next generation cryptography to protect US government information", including the protection of "classified or mission critical national security information" [57, 42].

By implicitly authenticated DHKE, we mean DHKE protocols whose communication is identical to the basic DH protocol, yet they are implicitly authenticated by the sole ability of the parties to compute the resultant session key [42]. IA-DHKE is initially suggested in [47], which then triggered a list of subsequent (ad-hoc) designs of IA-DHKE protocols. The (H)MQV protocols marked the great milestones of IA-DHKE developments. In particular, the first formal analysis of IA-OAKE, say the HMQV protocol, within the Canetti-Krawczyk framework (CK-framework, for short) is conducted in [42], which is particularly helpful in understanding the protocol design insights and the parameter choices. Though seemingly conceptually simple, the (H)MQV protocols are actually very delicately designed, in the sense that slight change in the design structure or parameters can lose provable security or even be totally insecure. Indeed, despite its seemingly conceptual simplicity, designing "*sound*" and "*right*" key-exchange protocols turns out to be extremely error prone and can be notoriously subtle. Also, the analysis of even a simple cryptographic protocol in intricate adversarial settings like the Internet can be a luxury and dauntingly complex task [12, 42]. The reason for this is the high system complexity and enormous number of subtleties surrounding the design, definition and analysis of key-exchange protocols.

[*] This work is an academic version of [66, 67], where the protocols presented in this work originally appeared there.

[†] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. andrewcyao@tsinghua.edu.cn

[‡] Software School, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn

The provable security of HMQV, assuming exposed DH-components and DH-exponents, additionally relies on the non-standard KEA assumption [42] (for the most often cases of $\hat{A} \neq \hat{B}$ or $X \neq Y$). Developing provably secure IA-DHKE protocols *resilient to the leakage of DH components and exponents in advance*, which is as efficient as (H)MQV while removing the additional KEA assumption (or relying on the KEA assumption as minimal as possible), was left as an interesting open question. For key-exchange protocols, besides the need of sound security, protocol efficiency, privacy protection, and computational fairness also play a critical role in protocol evaluation and standard selection. In these dimensions, (H)MQV also leaves much space to improve.

- (H)MQV does not support offline pre-computing parts of the shared DH-secret, and does not support leaving some expensive exponentiation computations to untrusted computing devices, which quite limits the deployment of (H)MQV with power-limited devices (e.g., smart-cards, mobile phones, etc). In particular, achieving online-optimal (say, only one online exponentiation) provably secure IA-DHKE remains an open question in the literature. Notice that, as (H)MQV is commonly viewed as the state-of-the-art in the integrity of security and efficiency, even minor efficiency improvement while remaining the provable security is challenging.
- The design of (H)MQV considers little in privacy protection. In this work, we focus on the privacy-preserving properties of deniability and post-ID computability, which have served as crucial evaluation criteria in evaluating and selecting the IKEv2 standard [40, 41].
- In this work, we identify some asymmetry or unfairness in session-key computation of (H)MQV between a malicious player and an honest player. Specifically, a malicious player can pay much lesser computational resource in computing the session-key, and can even set the session-key to be some pre-determined or publicly computable values.

It is thus much desirable to present new IA-DHKE protocols, which preserve the advantages of (H)MQV in provable security and efficiency (or, perform even better) while overcoming the above mentioned disadvantages. But, as mentioned, the protocol structure of HMQV is quite delicate and sensitive to the security analysis in the CK-framework, and even a slight change in protocol structure or parameters can lose provable security or even be totally insecure. We do not know how to directly modify the (H)MQV protocol structure to overcome the above mentioned disadvantages while preserving its provable security and efficiency. This thus calls for some new protocol structures with new design novelty and rationales.

In this work, inspired by the design of deniable Internet key-exchange [68], we start with investigating practical mechanisms in the random oracle (RO) model, referred to as non-malleable joint proof-of-knowledge (NMJPOK) for presentation simplicity, for proving DH-knowledges, say both the secret-key and the DH-exponent, *jointly* and *non-malleably* in concurrent settings like the Internet. In light of this line of investigations and inspired by the security analysis of HMQV, we develop a new family of practical IA-DHKE protocols, which consists of three protocols referred to as OAKE,[1] single-hash OAKE (sOAKE) and robust OAKE (rOAKE). For presentation simplicity, we refer to the newly developed DHKE protocols as (s,r)OAKE. Sometimes, we also refer to (s)OAKE (resp., (r)OAKE) as the protocols of OAKE and sOAKE (resp., rOAKE).

We then make detailed comparisons between (s,r)OAKE and (H)MQV, which shows that the new protocols outperform HMQV in most aspects. Detailed comparisons are listed in Section 4 and Table 1 (page 11), after motivating the design rationales and building tools and after presenting the detailed OAKE specifications. Similar to HMQV, despite its seemingly conceptual simplicity, the OAKE protocol family was also very delicately designed and chosen among various potential protocol variants and protocol parameters, toward an optimal balance among efficiency, security, privacy, fairness and easy deployment. Along the way, guided by our new design rationales, we also introduce and discuss some new concept, say session-key computational fairness (as a complement to session-key security),

[1] There are two acronym interpretations of OAKE. One interpretation is: (Online) Optimal (implicitly) Authenticated (Diffie-Hellman) Key-Exchange. Another interpretation is: (Toward) Optimally-balanced (implicitly) Authenticated (Diffie-Hellman) Key-Exchange (in the integrity of protocol efficiency, security, privacy, fairness and easy deployment).

to the literature. To the best of our knowledge, our work is the first that formally treats the issue of computational fairness for cryptographic protocols, which is **in particular** significantly different from the concept of "complete fairness" studied in the literature of secure multi-party computation (SMC).

2 Preliminaries

If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots; r)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. If S is a finite set then $x \leftarrow S$, sometimes also written as $x \in_{\mathrm{R}} S$, is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement.

Let G' be a finite Abelian group of order N, G be a subgroup of prime order q in G'. Denote by g a generator of G, by 1_G the identity element, by $G \setminus 1_G = G - \{1_G\}$ the set of elements of G except 1_G and by $t = \frac{N}{q}$ the cofactor. In this work, we use multiplicative notation for the group operation in G'. We assume the computational Diffie-Hellman (CDH) assumption holds over G, which roughly says that given $X = g^x, Y = g^y \leftarrow G$ (i.e., each of x and y is taken uniformly at random from Z_q) no probabilistic polynomial-time (PPT) algorithm can compute $CDH(X, Y) = g^{xy}$ with non-negligible probability.

Let $(A = g^a, a)$ (resp., $(X = g^x, x)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{A}$, and $(B = g^b, b)$ (resp., $(Y = g^y, y)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{B}$, where a, x, b, y are taken randomly and independently from Z_q^*. (H)MQV is recalled in Figure 1 (page 6), and the (H)MQV variants are recalled in Appendix A, where on a security parameter k H_K (resp., h) is a hash function of k-bit (resp., l-bit) output and l is set to be $|q|/2$. *Throughout this work, we assume that the underlying PKI requires no proof-of-knowledge (POP) or proof-of-possession (POP) of secret-key during key registration, but the CA will check the (non-identity) sub-group membership of registered public-keys (i.e., make sure that the registered public-keys are in $G \setminus 1_G$).*

The Gap Diffie-Hellman (GDH) assumpiton [58] roughly says that the CDH assumption holds even if the CDH solver is equipped with a decisional Diffie-Hellman (DDH) oracle for the group G and generator g, where on arbitrary input $(U, V, Z) \in G^3$ the DDH oracle outputs 1 if and only if $Z = CDH(U, V)$.

Informally speaking, the knowledge-of-exponent assumption (KEA) assumption says that, suppose on input $(g, C = g^c)$, where c is taken uniformly at random from Z_q^*, an efficient (say, probabilistic polynomial-time) algorithm $\mathcal{A}$ outputs $(Y, Z = Y^c) \in G^2$, then the discrete logarithm y of $Y = g^y$ can be efficiently extracted from the input (g, C) and the random coins used by $\mathcal{A}$. The KEA assumption is derived from the CDH assumption, and is a *non-black-box* assumption by nature [7]. The KEA assumption was introduced in [19], and has been used in many subsequent works (e.g., [35, 8, 7, 21, 42, 20, 22], etc). In particular, the KEA assumption plays a critical role for provable deniability of authentication and key-exchange (e.g., [21, 42, 22]). More details are referred to Appendix B.

Brief description of the CK-framework. In the CK-framework for a DHKE protocol, a session run at the side of player $\hat{A}$ with a peer player $\hat{B}$, where the outcoming (resp., incoming) DH-component is X (resp., Y) and $\hat{A}$ and $\hat{B}$ may be the same player, is identified as $(\hat{A}, \hat{B}, X, Y)$. In each session, a party can be activated as the role of either initiator (who sends the first DH-component) or responder (who sends the second DH-component). The session $(\hat{B}, \hat{A}, Y, X)$ (it it exists) is said to be matching to $(\hat{A}, \hat{B}, X, Y)$, *if the two players have matching player roles in these two sessions, i.e., if $\hat{A}$ is the initiator (resp., responder) then $\hat{B}$ is the responder (resp., initiator).*[2]

A CMIM adversary $\mathcal{A}$ controls all the communication channels among concurrent session runs of the KE protocol. In addition, $\mathcal{A}$ is allowed access to secret information via the following three types of queries: (1) state-reveal queries for ongoing incomplete sessions; (2) session-key queries for completed sessions; (3) corruption queries upon which all information in the memory of the corrupted parties will

[2] The requirement of matching roles was only implicitly assumed in the definition and security analysis in [42]. As explicitly observed in [18], without explicitly making this requirement, no DHKE protocol where session-key derivation is dependent of players' roles, just like HMQV and the OAKE family, can satisfy the completeness requirement, i.e., two matching sessions should output the same session-key.

be leaked to $\mathcal{A}$. A session $(\hat{A}, \hat{B}, X, Y)$ is called *exposed*, if it or its matching session $(\hat{B}, \hat{A}, Y, X)$ suffers from any of these three queries.

The session-key security (SK-security) within the CK-framework is captured as follows: for any complete session $(\hat{A}, \hat{B}, X, Y)$ adaptively selected by $\mathcal{A}$, referred to as the *test session*, as long as it is unexposed it holds with overwhelming probability that (1) the session-key outputs of the test session and its matching session are identical; (2) $\mathcal{A}$ cannot distinguish the session-key output of the test session from a random value. At a high level, the SK-security essentially says that a party that completes a session has the following guarantees [15]: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the *unexposed* peer completes a matching session then the two parties have the same shared key. The reader is referred to [15] for the detailed description of the CK-framework.

3 Design of OAKE: Motivation, Discussion and Specification

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a Diffie-Hellman protocol $\langle \hat{A}, \hat{B} \rangle$ are run concurrently over an asynchronous network like the Internet. To distinguish concurrent sessions, each session run at the side of an uncorrupted player is labeled by a tag, which is the concatenation, in the order of session initiator and then session responder, of players' identities/public-keys and DH-components available from the session transcript. A session-tag is complete if it consists of a complete set of all these components.

In this work, we study the mechanisms, in the random oracle (RO) model, for *non-malleably* and *jointly* proving the knowledge of both b and y w.r.t. a challenge DH-component X between the prover $\hat{B}$ (of public-key $B = g^b$ and DH-component $Y = g^y$) and the verifier $\hat{A}$ (who presents the challenge DH-component $X = g^x$), where $b, y, x \leftarrow Z_q^*$. For presentation simplicity, such protocol mechanism is referred to as $JPOK(b, y)$. Moreover, we look for solutions of $JPOK_{(b,y)}$ such that $JPOK_{(b,y)}$ can be efficiently computed with one single exponentiation by the knowledge prover. Note that the tag for a complete session of $JPOK_{(b,y)}$ is $(\hat{A}, \hat{B}, B, X, Y)$. Throughout this work, unless otherwise specified, we use a hash function $h : \{0,1\}^* \to \{0,1\}^l \setminus \{0\} \subseteq Z_q^*$ (in the unlikely case that $h(x) = 0$ for some x, the output of $h(x)$ can be defined by default to be a value in $Z_q^* \setminus \{0,1\}^l$), which is modeled as a random oracle, and we denote by the output length, i.e., l, of h as the security parameter.

Our starting point is the JPOK mechanism proposed in [68] for deniable Internet key-exchange (IKE): $JPOK_{(b,y)} = h(\hat{A}, A, \hat{B}, B, Y, X, X^b, X^y)$ w.r.t. a random DH-component challenge X from $\hat{A}$. This JPOK mechanism is shown to be sound in the RO model [68], but is less efficient and has no way to be used for IA-DHKE. In order to further improve its efficiency toward a building tool for IA-DHKE, one naive solution is to just set $JPOK_{(b,y)} = X^b \cdot X^y = X^{b+y}$. But, such a naive solution is totally insecure, for example, an adversary $\mathcal{A}$ can easily impersonate the prover $\hat{B}$ and pre-determine $JPOK_{(b,y)}$ to be 1_G, by setting $Y = B^{-1}$. The underlying reason is: $\mathcal{A}$ can malleate B and Y into X^{y+b} *by maliciously correlating the exponents of y and b*, but actually without knowing either of them. A further remedy of this situation is to mask the exponents b and y by some random values. In this case, the proof is denoted as $JPOK_{(b,y)} = X^{db+ey}$, where d and e are random values (e.g., $d = h(X, \hat{B})$ and $e = h(Y, \hat{A})$). The intuition with this remedy solution is: since d and e are random values, db and ey are also random and independent. This intuition however turns out also to be wrong *in general*. With the values $d = h(B, \hat{A})$ and $e = h(X, \hat{B})$ as an illustrative example, after receiving X an adversary $\mathcal{A}$ can generate and send $Y = B^{-d/e}$, and in this case $JPOK_{(b,y)} = X^{db+ey} = 1_G$. This shows that masking b and y by random values is also not sufficient for ensuring the non-malleability of $JPOK_{(b,y)}$. The key point here is that the values db and ey are *not* necessarily *independent*, and thus a malicious prover can still make the values db and ey correlated. Thus, one key principle is to ensure the values db and ey to be "computationally independent", no matter what a malicious prover does, which is referred to as inside non-malleability. In addition, as JPOK may be composed with other protocols in practice, another principle is that the JPOK provided by one party in a session should be bounded to that session, in the sense that the JPOK should not be malleated *into* or *from* other sessions, which is referred to as outside non-malleability. This is captured by the tag-based self-seal (TBSS) definition (see Definition C.2), by requiring the complete session-tag to be committed to JPOK. Informally speaking, we say a JPOK protocol is non-malleable joint proof-of-knowledge (NMJPOK), if it enjoys both inside non-malleability and outside non-malleability.

This line of investigations bring us to the following two candidates for NMJPOK of both b and y w.r.t. random challenge X, under the preference of on-line efficiency and minimal use of hashing. More details are referred to Appendix C.

- NMJPOK: $NMJPOK_{(b,y)} = X^{db+ey}$, where $d = h(B, X)$ and $e = h(X, Y)$;
- Single-hash NMJPOK (sNMJPOK): $sNMJPOK_{(b,y)} = X^{db+ey}$, where $d = 1$ and $e = h(B, X, Y)$.

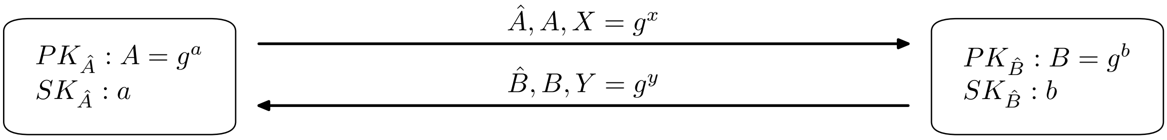
Below, we provide some informal justifications of $NMJPOK$ and $sNMJPOK$, by avoiding introducing and employing some cumbersome terminologies for easier interpretation. Formal treatments are referred to Appendix C. Informally speaking, the underlying rationale of $NMJPOK_{(b,y)}$ is: given a random challenge X, no matter how a malicious $\hat{B}$ chooses the values $Y = g^y$ and $B = g^b$ (where y and b can be arbitrarily correlated), it actually has no control over the values db and ey in the RO model. That is, by the birthday paradox, it is infeasible for a malicious $\hat{B}$ to set db (resp., ey) to some predetermined value, which may be determined by ey (resp., db) via some predetermined polynomial-time computable relation $\mathcal{R}$, in the RO model in order to make the values db and ey correlated with non-negligible probability. Thus, given a random challenge X, it is infeasible for a malicious $\hat{B}$ to output $B = g^b$ and $Y = g^y$ such that the values db and ey satisfy some predetermined relation $\mathcal{R}$ with non-negligible probability in the RO model.

The situation with $sNMJPOK_{(b,y)}$ is a bit different. Though as in $NMJPOK_{(b,y)}$, the malicious $\hat{B}$ is infeasible to set ey to some predetermined value, $\hat{B}$ can always set the value $db = b$ at its wish as $d = 1$ for $sNMJPOK_{(b,y)}$. But, $\hat{B}$ is still infeasible to set the value b correlated to $ey = h(B, X, Y)y$, particularly because the value B is put into the input of (i.e., committed to) e. Specifically, for any value $B = g^b$ set by $\hat{B}$, with the goal of making b and ey correlated, the probability that the values $ey = h(B, X, Y)y$ and b satisfy some predetermined polynomial-time computable relation $\mathcal{R}$ is negligible in the RO model. In particular, by the birthday paradox, the probability that $\Pr[b = f(ey)]$ or $\Pr[f(b) = ey]$, where f is some predetermined polynomial-time computable function (that is in turn determined by the predetermined relation $\mathcal{R}$), is negligible in the RO model, no matter how the malicious $\hat{B}$ does.

Note that $NMJPOK_{(b,y)} = X^{db+ey} = (B^d Y^e)^x$, where $d = h(B, X)$ and $e = h(X, Y)$, actually can be used to demonstrate the knowledge of x. The key observation now is: in order for $\hat{A}$ to additionally prove the knowledge of its secret-key a, we can multiply X^{db+ey} by another POK Y^{ca} for $c = h(A, Y)$. This yields $K_{\hat{A}} = B^{dx} Y^{ca+ex} = A^{cy} X^{db+ey} = K_{\hat{B}}$, where $K_{\hat{A}}$ (resp., $K_{\hat{B}}$) is computed by $\hat{A}$ (resp., $\hat{B}$) respectively. As we aim for secure DHKE protocols in concurrent settings like the Internet, in order to ensure outside non-malleability in accordance with the TBSS definition discussed in Appendix C, we let the values $K_{\hat{A}}$ and $K_{\hat{B}}$ commit to the complete session tag by putting users' identities into the inputs of d and/or e, which particularly ensures the key-control property of [45] for DHKE. The variant derived from sNMJPOK is referred to as single-hash OAKE (sOAKE). Robust OAKE (rOAKE) is derived from OAKE by additionally multiplying the value g^{ab} into the shared DH-secret $K_{\hat{A}} = K_{\hat{B}}$, which is guided by the analysis of HMQV for the need of removing the additional non-standard KEA assumption for rOAKE. All these three protocols are depicted in Figure 1.

Note that the output length of h, i.e., l, is set to be $|q|/2$ in (H)MQV, but approximately $|q|$ in (s,r)OAKEs. In particular, with the OAKE protocol family, h and H_K (that is used for deriving the session-key K) can be identical. Also note that, for OAKE and sOAKE, referred to as (s)OAKE for simplicity, $\hat{A}$ (resp., $\hat{B}$) can offline pre-compute X and B^{dx} (resp., Y and A^{cy}), while for rOAKE $\hat{A}$ (resp., $\hat{B}$) can offline pre-compute X and B^{a+dx} (resp., Y and A^{b+cy}). Some more variants of (s,r)OAKE are given in Appendix D. Here, we also highlight the TBSS property of (s,r)OAKE in the RO model: given any complete session tag $(\hat{A}, A, \hat{B}, B, X, Y)$ and any $\alpha \in G \setminus 1_G$, $\Pr[K_{\hat{A}} = K_{\hat{B}} = \alpha] \leq \frac{1}{2^l - 1}$, where the probability is taken over the choice of the random function of h (see more discussions on TBSS in Appendix C).[3]

[3] Actually, if h is modeled to be an RO from $\{0,1\}^*$ to Z_q^*, $K_{\hat{A}} = K_{\hat{B}}$ is distributed uniformly at random over $G \setminus \{1_G\}$.



(H)MQV: $K_{\hat{A}} = (YB^e)^{x+da}, K_{\hat{B}} = (XA^d)^{y+eb}, K = H_K(K_{\hat{A}}) = H_K(\hat{B})$

MQV: $d = 2^l + (X \bmod 2^l), e = 2^l + (Y \bmod 2^l), l = |q|/2$

HMQV: $d = h(X, \hat{B}), e = h(Y, \hat{A}), l = |q|/2$

(s,r)OAKE: $K_{\hat{A}} = B^{fa+dx}Y^{ca+ex}$, $K_{\hat{B}} = A^{fb+cy}X^{db+ey}$, $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$

OAKE: $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$, $e = h(X, Y)$, $f = 0$, $l \approx |q|$

sOAKE: $c = d = 1$, $e = h(\hat{A}, A, \hat{B}, B, X, Y)$, $f = 0$, $l \approx |q|$

rOAKE: $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$, $e = h(X, Y)$, $f = 1$, $l \approx |q|$

Figure 1: Specifications of (H)MQV and (s,r)OAKE

On embedded subgroup tests in (s,r)OAKE. The basic technique to check the DH-component, e.g. X, is in G is to verify $X^q = 1_G$ (and $X \in G' \backslash 1_G$) that needs performing one modular exponentiation. But, if the cofactor t is small, e.g., $G' = Z_N^*$ such that $N = 2q+1$ or G is the subgroup of an elliptic curve over a finite field (in this case the cofactor t is usually a small constant), the subgroup test of X can be essentially reduced to: (1) check $X \in G'$; (2). $X^t \neq 1_G$. In general, checking $X \in G'$ and $X^t \neq 1_G$ guarantees that X is not in a (small) subgroup of G' with the order that is a factor of t, but it does not fully guarantee $X \in G$ (e.g., considering that $X = -g^x$). This leads to the following (s,r)OAKE variant with embedded subgroup tests, in which the values $K_{\hat{A}}, K_{\hat{B}}$ are set to be: $K_{\hat{A}} = B^{aft+dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{bft+cyt}X^{dbt+eyt}$, where $f = 0$ for (s)OAKE and $f = 1$ for rOAKE. The subgroup test is performed as follows: each player first verifies that its peer's DH-component is in G', and then acts in accordance with one of the following two cases.

Case-1. If $B^{aft+dxt}$ and $Y^{cat+ext}$ (resp., $A^{bft+cyt}$ and $X^{dbt+eyt}$) are computed separately, *particularly when $B^{aft+dxt}$ (resp., $A^{bft+cyt}$) is offline pre-computed by $\hat{A}$ (resp., $\hat{B}$)*, $\hat{A}$ (resp., $\hat{B}$) checks that $Y^{cat+ext} \neq 1_G$ (resp., $X^{dbt+eyt} \neq 1_G$);

Case-2. In case of no separate computation, $\hat{A}$ (resp., $\hat{B}$) verifies $K_{\hat{A}} \neq 1_G$ (resp., $K_{\hat{B}} \neq 1_G$). Note that the checking of $K_{\hat{A}} \neq 1_G$ and $K_{\hat{B}} \neq 1_G$, as done in MQV, does not *fully* guarantee $X^t \neq 1_G$ or $Y^t \neq 1_G$, but it still provides reasonable assurance in the elliptic curve setting.

We remark that the embedded subgroup test in Case-1, well supported by (s,r)OAKE, provides stronger security guarantee than that in Case-2 as done in (H)MQV. Note that (H)MQV cannot offline pre-compute the values B^e and A^d to ease the more robust Case-1 embedded subgroup test. We note that the damage caused by ignoring the subgroup test of peer's DH-component (but still with the supergroup G' membership check) can be much relieved (and even waived), if the ephemeral private values generated within the protocol run are well-protected. More notes on the subgroup test, and on the ephemeral private values that can be exposed to adversary, are referred to Appendix E.

4 Discussions on Advantageous Features and Security

Efficiency advantages. The online computational complexity of (s,r)OAKE can be only 1 exponentiation at each player side (with embedded subgroup test), which is optimal for DHKE **and is important for deploying a DHKE protocol with power-limited devices like smart-cards, mobile-phones, etc)**. Specifically, the value B^{fa+dx} (resp., A^{fb+cy}) can be offline pre-computed by $\hat{A}$ (resp., $\hat{B}$). In comparison, (H)MQV cannot offline pre-compute the values B^e and A^d to improve online efficiency, and thus the online efficiency of (H)MQV is about 1.3 exponentiations by the simultaneous exponentiation techniques [49, 33, 24].

The total computational complexity of (s,r)OAKE is essentially the same as that of (H)MQV, with sOAKE being still slightly more efficient than HMQV. In particular, by the simultaneous exponentiation techniques [49, 33, 24], each player in (H)MQV and (s,r)OAKE performs about 1.3 exponentiations in computing $K_{\hat{A}}$ or $K_{\hat{B}}$. But, the computation of $K_{\hat{A}}$ (resp., $K_{\hat{B}}$) of HMQV is still slightly more inefficient than that of sOAKE with a single hash. For example, to compute $K_{\hat{A}}$, besides the same other operations needed for simultaneous exponentiations, HMQV (resp., sOAKE) needs to compute $\{d, e, x + da, e(x + da)\}$ (resp., only $\{e, a + xe\}$). To the best of our knowledge, the sOAKE protocol is the most efficient provably secure IA-DHKE protocol to date. As mentioned, due to the state-of-the-art nature and highly intensive study of (H)MQV, even slight efficiency improvement can be challenging.

On the same subgroup order q, (s,r)OAKE ensures more robust resistance to collision attacks against the hash function h than HMQV, as the output length of h, i.e., l, is set to be $|q|/2$ for HMQV but $|q|$ for (s,r)OAKE. To strengthen the resistance to birthday-type collision attacks against h (in case of pre-computed DH-components exposed prior to the sessions involving them and a large number of sessions), HMQV needs some changes (as specified in [42] and also discussed in Appendix G), and some standards mandate larger subgroups (e.g., $|q| = 255$ in [57]) to use for HMQV. However, in memory-restricted environments (like smart-cards or other portable electronic tokens), subgroup size is an influential parameter in favor of a given algorithmic solution.

Reasonable deniability. For key-exchange protocols, both security and privacy are desired, which would also have been being one of the major criteria underlying the evolution of a list of important industrial standards of KE protocols (e.g., Internet key-exchange). Among privacy concerns, deniability is an essential privacy property, and has always been a central concern in personal and business communications, with off-the-record communication serving as an essential social and political tool [21, 22, 25]. The reader is referred to [21, 22, 25] for a list of scenarios where deniability is desirable. (Needless to say, there are special applications where non-repudiable communication is essential, but this is not the case for most of our nowaday communications over Internet [21, 22, 25] where deniable authentication is much more desirable than non-repudiable authentication.)

A 2-round implicitly authenticated DHKE protocol is defined to be of reasonable deniability, if the session-key can be computed merely from the ephemeral DH-exponents without involving any player's static secret-key. Note that we cannot count on implicitly authenticated DHKE, like (H)MQV and (s,r)OAKE, to enjoy full-fledged deniability (zero-knowledge). It is clear that (s)OAKE enjoys reasonable deniability, as the session-key of (s)OAKE can be computed merely from the DH-exponents x and y, which is useful to preserve privacy for both protocol players. Note that (H)MQV and also the rOAKE protocol are not reasonably deniable, as the value g^{ab} is involved in the session-key computation, and thus the use of the session-key of (H)MQV and rOAKE can be traced back to the group of the two participating players.

Modular, parallel and post-ID computability. First note that B^{fa+dx}, Y^{ca+ex} and the explicit sub-group test Y^q by $\hat{A}$ (resp., A^{fb+cy}, X^{db+ey} and X^q by $\hat{B}$) can be computed in a parallel, modular and post-ID way, which allows for various trade-offs among security, privacy and efficiency for the deployment of (s,r)OAKE in practice. Specifically, the offline pre-computability of B^{fa+dx} and A^{fb+cy} eases more efficient explicit subgroup test by computing Y^{ca+ex} and Y^q (resp., X^{db+ey} and X^q) *in parallel* that amounts to about 1.2 exponentiations. Also, as clarified, offline pre-computability of A^{fb+cy} (resp., B^{fa+dx}) allows the above more robust Case-1 embedded subgroup test of $X^{dbt+ext}$ (resp., $Y^{cat+ext}$). Further observe that, for OAKE and rOAKE ((r)OAKE, in short), Y^{ca+ex} (resp., X^{db+ey}) can be computed before learning peer's identity and public-key information. Such a post-ID computability, as desired with IKEv2 [40, 41], is useful for privacy preservation [16]. (H)MQV does not support such offline pre-computability and post-ID computability.

Security with public computation. The work [44] proposed the public computation model for KE protocols, where an entity (performing a run of KE-protocol) is split into two parts: a trusted authentication device (which enforces the confidentiality of the authentication data), and an untrusted computing device (in which some computing operations are *publicly* carried out). This allows to use an authentication device with little computing power, and to make computing devices independent from users [44]. Some concrete applications suggested in [44] are: (1) Mobile phones include smart cards which store the user authentication data; the handsets themselves are the computing devices. (2) PCs (corresponding to the computing device) equipped with a crypto token (corresponding to the authentication device) have a lot more computing power than the token itself, but may be plagued by spyware or virus. (H)MQV does not support deployment with such public computation as shown in [44], while (s,r)OAKE well supports deployment in this model (see details in Appendix H).

Specifically, the natural split of authentication computation and public computation for (s,r)OAKE is as follows, with the computation of $\hat{B}$ as an example: (1) The authentication device generates (y, Y) and possibly A^{fb+cy} (in case the authentication device has learnt the peer identity $\hat{A}$), and then forwards Y and possibly A^{fb+cy} to the computation device. (2) After getting X from the computation device, the authentication device computes $s = db + ey$, and then forwards s to the computation device. (3) After getting s from the authentication device, the computation device computes $K_{\hat{B}} = A^{fb+cy}X^s$ and the session-key, and then communicates with $\hat{A}$ with the session-key. *Note that $y, Y, c, d, A^{fb+cy}, db$ can be offline pre-computed by the authentication device, and the authentication device needs only online computing ey and s. Also, the computation device essentially needs to compute only one exponentiation X^s.* More details are referred to Appendix H.1.

Ease deployment with lower-power devices. Online optimal efficiency and supporting public computation, as clarified above, make (s,r)OAKE more desirable for being deployed with devices of limited computational ability in hostile computing environments. Also, (s,r)OAKE allows smaller parameter $|q|$ than HMQV in resistance to collision attacks against h, which is important for deployment with memory-restricted devices (like smart-cards or other portable electronic tokens).

Minimal setup. (s,r)OAKE does not mandate proof of possession/knowledge (POP/K) of secret-key during public-key registration, while POP/K is now commonly assumed for MQV. POP/K is explicitly abandoned in HMQV, however as we shall see, there exists a way to maliciously *asymmetrically* compute the session-key of HMQV *without knowing either static secret-key or ephemeral DH-exponent.*

Security in the CK-framework. For the security of (s,r)OAKE in the CK-framework, we have the following theorems:

Theorem 4.1 *The OAKE and sOAKE (resp., rOAKE) protocols, with the exposed values of offline pre-computed DH-components, DH-exponents and the DH-secrets of one's DH-component and its peer's public-key, say A^{cy} and B^{dx} (resp., only offline pre-computed DH-components, DH-exponents), are secure in the CK-framework under the following assumptions:*

Case-1. *The GDH assumption, w.r.t. any test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ where $\hat{A} \neq \hat{B}$.*

Case-2. *Both the GDH assumption and the KEA assumption (resp., merely the GDH assumption), w.r.t. any test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ where $\hat{A} = \hat{B}$ but $X_0 \neq Y_0$.*

Case-3. *The CDH assumption, w.r.t. any test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ where $\hat{A} = \hat{B}$ and $X_0 = Y_0$.*

Note that the security of rOAKE does not rely on the non-standard KEA assumption, but allows more limited secrecy exposure. Specifically, for provable security of rOAKE, the values A^{b+cy} and B^{a+dx} should be well protected and should not be exposed in case they are offline pre-computed. The actual security proofs are quite lengthy and technical. As we focus on conveying conceptual messages with the main text, proof details are referred to Appendix G. In comparison, with exposed DH-components and DH-exponents, the security of HMQV relies on both the GDH assumption and the KEA assumption for both Case-1 and Case-2. More discussions on the security of (s,r)OAKE vs. HMQV are given in Appendix F. Due to space limitation, more discussions on security of (s,r)OAKE beyond the CK-framework are also referred to Appendix H.

5 Computational Fairness: A New Perspective to Cryptographic Protocols

In this work, we identify some asymmetry or unfairness in session-key computation of (H)MQV between a malicious player and an honest player. Specifically, a malicious player can pay much lesser computational resource in computing the session-key, and can even set the session-key to be some predetermined or publicly-computable values. This is demonstrated by a new kind of attacks, referred to as *exponent-dependent attacks* (EDA). We then discuss the implications and damages of this vulnerability, and then introduce a new concept called "(session-key) *computational fairness*" for (implicitly authenticated) DHKE protocols.

5.1 Exponent Dependent Attack

Given a value $X \in G$ for which the malicious player $\hat{A}$ (e.g., a client) does not necessarily know the discrete logarithm of X, $\hat{A}$ computes d and sets $A = X^{-d^{-1}} \cdot g^t$ where $t \in Z_q$ and $d = h(X, \hat{B})$ for HMQV or $d = 2^l + (X \mod 2^l)$ for MQV. Note that $XA^d = X(X^{-d^{-1}} \cdot g^t)^d = XX^{-1}g^{td} = g^{td}$, and the shared DH-secret now is $K_{\hat{A}} = (XA^d)^{y+eb} = g^{tdy}g^{tdeb} = Y^{td}B^{tde}$. We call such an attack exponent dependent attack. If $\mathcal{A}$ sets $t = 0$ then the shared DH-secret $K_{\hat{A}}$ is always 1_G. If $\mathcal{A}$ sets $t = d^{-1}$, then $K_{\hat{A}} = YB^e$. For all these two specific cases, the value $K_{\hat{A}}$ can be *publicly computed* (without involving any secret values). In any case, the computational complexity in computing the shared DH-secret by the malicious $\hat{A}$ is much less than that by its peer $\hat{B}$, which clearly indicates some unfairness. In general, the malicious $\hat{A}$ can honestly generate its public-key $A = g^a$ and compute the session-keys, thus explicitly requiring POP/K of secret-key during public-key registration and explicit key-confirmation and mutual authentication (as required by the 3-round (H)MQV) do not prevent the above attacks. As there are many choices of the value t by the adversary in different sessions, explicitly checking whether the shared DH-secret is YB^e also does not work. The above attacks can also be trivially modified (actually simplified) to be against the one-round HMQV variant. *We stress that such attacks do not violate the security analysis of HMQV in [42], as they are beyond the CK framework.*

We note that MQV (with embedded subgroup membership test of peer's DH-component) explicitly checks the shared DH-secret is not 1_G, and thus the attack with $t = 0$ does not work against MQV. But, for (H)MQV with explicit subgroup tests of peer's public-key and DH-component, whether still checking the shared DH-secret is 1_G is however unspecified. In particular, the basic version of HMQV [42] does not check whether the shared DH-secret is 1_G or not, and POP/K of secret-keys is explicitly abandoned in HMQV. We also note the version of HMQV proposed in [43] does check and ensures the shared DH-secret is not 1_G. But, (H)MQV does not resist the above attacks with $t \neq 0$.

Besides asymmetric computation, such a vulnerability also allows more effective DoS attacks. Though an adversary can send arbitrary messages to an honest party (say, player $\hat{B}$ in the above attacks) to issue DoS attacks, which however can be easily detected by the authentication mechanism of (the 3-round version of) (H)MQV. But, with our above attacks, the honest player $\hat{B}$ is hard to distinguish and detect an attack from an honest execution of (H)MQV.

5.2 Formulating Computational Fairness

This motivates us to introduce a new notion for DHKE, called session-key computational fairness. For any complete session-tag Tag of a key-exchange protocol among n users $\{U_1, \cdots, U_n\}$ where $n \geq 2$, we first identify *dominant-operation values* w.r.t. Tag and each user U_i, $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m, m \geq 2$, which are specified to compute the session-key K by the honest player U_i for a complete session specified by the complete session-tag Tag, where G_k, $1 \leq k \leq m$ is the range of V_k^i. Specifically, $K = F_K(V_1^i, \cdots, V_m^i, Tag)$, where K is the session-key output by user U_i, F_K is some polynomial-time computable function (that is defined by the session-key computation specified for honest players). We remark that dominant operations are specific to protocols, where for different key-exchange protocols the dominant operations can also be different.

Definition 5.1 (non-malleable independence) *For the dominant-operation values, $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m$, $m \geq 2$ and $1 \leq i \leq n$, w.r.t. a complete session-tag Tag on any sufficiently large security parameter l, we say $V_1^i, \cdots, V_m^i$ are computationally (resp., statistically) non-malleably independent, if for any polynomial-time computable (resp., any power unlimited) relation/algorithm $\mathcal{R}$ (with components drawn from $G_1 \times \cdots \times G_m \times \{0,1\}^*$) it holds that the following quantity is negligible in l:*

$$\left|\Pr[\mathcal{R}(V_1^i, \cdots, V_m^i, Tag) = 1] - \Pr[\mathcal{R}(R_1, \cdots, R_m, Tag) = 1]\right|,$$

where $R_i, 1 \leq i \leq m$ is taken uniformly and independently from G_i, and the probability is taken over the random coins of $\mathcal{R}$ (as well as the choice of the random function in the random oracle model [10]).

Definition 5.2 ((session-key) computational fairness) *We say a key-exchange protocol enjoys session-key computational fairness w.r.t. some pre-determined dominant operations, if for any complete session-tag Tag on any sufficiently large security parameter l, the session-key computation involves the same number of (statistically or computationally)* non-malleably independent *dominant-operation values for any (whether honest or malicious) user U_i, $1 \leq i \leq n$.*

Note that the notion of "(session-key) computational fairness" is defined w.r.t. some predetermined dominant operations that are uniquely determined by the protocol specification. We remark that it is the task of the protocol designer to specify the dominant operations, with respect to which computational fairness will be provably proved. Some comments and clarifications about the above formulation of computational fairness are in place (more details are referred to Appendix I):

Implication against malicious players. Though session-key computational fairness, as well as non-malleable independence, is defined w.r.t. any complete session tag, it actually ensures that: for any subset of malicious polynomial-time players, no matter how they do, *by the birthday paradox* it is infeasible from them to make the values $V_1^i, \cdots, V_m^i$ (involved in session-key computation) correlated under any predetermined polynomial-time computable relation for any $i, 1 \leq i \leq n$. In particular, the non-malleable independence definition ensures that, by the birthday paradox, for any successfully finished session among a set of (malicious and honest) players, no matter how the malicious players collude, it holds that: for any $i, 1 \leq i \leq n$ and for any values $(\alpha_1, \cdots, \alpha_m) \in G_1 \times \cdots \times G_m$, the probability that $\Pr[V_k^i = \alpha_i]$ is negligible for any $k, 1 \leq k \leq m$ and any $i, 1 \leq i \leq n$.

Why require dominant-operation values to be "non-malleably independent"? The reason is that, without such a requirement, as shown by our concrete EDA attacks, an adversary can potentially set these values maliciously correlated such that the session-key can be computed in a much easier way (than the way specified for honest players) *even without knowing any secrecy corresponding to the dominant-operation values.*

Why only require "involved" dominant-operation values? The reason we only require the dominant-operation values *involved* (rather than computed) in session-key computation is that, there can be multiple different ways to compute the session-key from dominant-operation values. With the function $F_K(X = g^x, Y = g^y) = X \cdot Y$ as an example, one can compute two separate exponentiations X and Y and then compute the session-key, but one can also use the simultaneous exponentiations technique to compute $X \cdot Y$ with only about 1.3 exponentiations. Furthermore, there are a number of different methods for simultaneous exponentiations with (slightly) varying computational complexities [49, 33, 24].

Session-key computational fairness vs. non-malleable joint proof-of-knowledge. For key-exchange protocols enjoying session-key computational fairness, if we view each non-malleably independent value as a proof-of-knowledge of the corresponding secrecy, the successful generation and use of session-key can be viewed as a non-malleable join proof-of-knowledge of secret values related to the non-malleable independent values. This further implies that a malicious player is infeasible to set the session-key to some values that can be publicly computed from the session transcript. In a sense, the notion of "self-sealed computational independence" in accordance with Definition C.1 (which is defined for NMJPOK) can be viewed as a special and weaker form of non-malleable independence defined here.

Computational fairness vs. complete fairness. The notion of "fairness" was intensively studied in the literature of secure multi-party computation (SMC) (see [31] for an overview of the various fairness notions considered in SMC). Informally speaking, a protocol is fair if either all the parties learn the output of the function, or no party learns anything (about the output). This property is also referred to as "complete fairness" (along with many variants), which mainly deals with prematurely adversarial aborting. To bypass some impossibility results on achieving fair SMC protocols with a majority of corrupted players, the work [29] introduced the notion of "resource fair SMC", which is still a variant of "complete fairness". Specifically, the "resource fairness" [29] captures "fairness through gradual release", which, roughly speaking, requires that the honest players and the adversary run essentially the same number of steps in order to obtain protocol output.

Casting "fairness through gradual release" into DHKE, it means that: players $\hat{A}$ and $\hat{B}$ gradually release their DH-components X and Y in sequential steps, so that both parties can output the session-key or both cannot. Clearly, the notions of "complete fairness" and "resource fairness" considered in the literature of SMC are significantly different from the session-key computational fairness considered in this work. Specifically, we assume both parties honestly send their DH-exponents, and computational fairness says that they then should invest essentially the same computational resources, i.e., no shortcut, in computing the session-key output. That is, our computational fairness is to capture the fairness between non-aborting players on the amount of computational resources invested for fulfilling some designated computing task, while "complete fairness" and its variant in the literature of SMC mainly deal with prematurely adversarial aborting.

On computational fairness of OAKE vs. (H)MQV. It is easy to check that, in the random oracle model, OAKE satisfies session-key computational fairness (w.r.t. *statistically* non-malleably independent dominant operation values), while (H)MQV does not as shown by the above concrete EDA attacks. We also observe that sOAKE and rOAKE also lose session-key computational fairness. More details are given in Appendix I.

6 Conclusion and Future Investigations

We conclude this work with a brief comparison between (s,r)OAKE and HMQV, and some suggestions for future investigations.

		OAKE	sOAKE	rOAKE	HMQV
Total efficiency		$1t_{se}$+$3t_m$+$3t_h$+$1t_a$	$1t_{se}$+$1t_m$+$1t_h$+$1t_a$	$1t_{se}$+$3t_m$+$3t_h$+$2t_a$	$1t_{se}$+$4t_m$+$2t_h$+$2t_a$
Online efficiency		$1t_e$	$1t_e$	$1t_e$	$1t_{se}$
Parallel subgroup test with pre-computation		$1.2t_e$	$1.2t_e$	$1.2t_e$	$1t_{se} + 1t_e$
Allowed secrecy exposure		(y, Y, A^{cy})	(y, Y, A^{cy})	(y, Y)	(y, Y)
Assumption	$\hat{A} \neq \hat{B}$	GDH	GDH	GDH	GDH+KEA
	$\hat{A} = \hat{B}, X \neq Y$	GDH+KEA	GDH+KEA	GDH	GDH+KEA
	$\hat{A} = \hat{B}, X = Y$	CDH	CDH	CDH	CDH
Support public computation		✓	✓	✓	×
Computationally fair		✓	×	×	×
Robust embedded subgroup test with pre-computation		✓	✓	✓	×
Reasonable deniability		✓	✓	×	×
Post-ID computability		✓	×	✓	×
Need change to resist collision attack on h?		No	No	No	Yes

Table 1: A brief comparison between (s, r)OAKE and HMQV

A brief comparison between (s,r)OAKE and HMQV is summarized in Table 1 (page 11). In Table 1, t_{se} stands for the time for performing one simultaneous exponentiations, t_e the time for performing one modular exponentiation, t_m the time for one modular multiplication over Z_q^*, t_h the time for performing one hashing operation, t_a the time for performing one modular addition over Z_q^*. For total efficiency, we only count the operations needed to compute the values $K_{\hat{A}} = K_{\hat{B}}$, as all other operations are the same for (s,r)OAKE and (H)MQV. For online efficiency, we only count the dominant operations for online computation parts of $K_{\hat{A}} = K_{\hat{B}}$ (while ignoring the non-dominant operations like t_m, t_a and t_h for presentation simplicity). By "parallel subgroup test with pre-computation", we mean that for (s,r)OAKE in case A^{fb+cy} is pre-computed, we can compute X^{db+ey} and X^q in parallel that amounts for about 1.2 exponentiations; (H)MQV does not support such a parallel operation. "Robust embedded subgroup test" means the more robust Case-1 subgroup test specified in Section 3 (page 6). The last item refers to the need to change the functions c, d, e, H_K or parameter l in order to resist birthday-type collision attacks on h, in case pre-computed DH-components are exposed prior to the sessions involving them and the number of sessions is large in the system. For example, suppose $|q| = 160$, we may hope the probability that collisions occur (by birthday attacks) should be at most 2^{-80} in such cases.

Finally, we end this work with some questions for future investigations:

- The design of (s,r)OAKE is based on the building tool of NMJPOK formulated in this work. An interesting direction is to develop (interactive) NMJPOK in the standard model, or non-interactive NMJPOK even in the random oracle model, and find their more applications.
- In this work, we mainly analyzed the security of (s,r)OAKE in the basic CK-framework. Another direction is to analyze the (s,r)OAKE in some more variants of the CK-framework introduced in recent years.
- For rOAKE, being merely based on the GDH assumption is achieved at the cost of losing reasonable deniability (and more restricted secrecy exposure). It is interesting to investigate the possibility of achieving provably-secure and *reasonably deniable* IA-DHKE merely based the GDH assumption (or other standard assumptions).
- We introduced the concept of session-key computational fairness, which can be viewed as a first step toward a formal treatment of computational fairness for cryptographic protocols. An interesting question is to achieve computationally-fair key-exchange protocols in the standard model, and to formulate and achieve more computationally-fair cryptographic protocols.

Acknowledgement. We thank Hugo Krawczyk for helpful comments and for referring us to [43].

References

[1] M. Abdalla, J.H An, M. Bellare, and C. Namprempre. From Identification to Signatures Via the FiatCShamir Transform: Necessary and Sufficient Conditions for Security and Forward-Security. *IEEE Transactions on Information Theory*, 54(8), 3631-3646 (2008). Preliminary version appeared in *EUROCRYPT* 2002.

[2] M. Abdalla, D. Catalano, C. Chevalier and D. Pointcheval. Password-Authenticated Group Key Agreement with Adaptive Security and Contributiveness. In Africacrypt'09, LNCS 5580, pages 254C271.

[3] American National Standard (ANSI) X9.42-2001. Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.

[4] American National Standard (ANSI) X9.42-2001. Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Elliptic Curve Cryptography.

[5] A.Bagherzandi, J.H. Cheon and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. ACM Conference on Computer and Communications Security 2008: 449-458

[6] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *N. Koblitz (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, Springer-Verlag, 1996.

[7] M. Bellare and A. Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *M. Franklin (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2004, LNCS 3152*, pages 273-289, Springer-Verlag, 2004.

[8] M. Bellare and A. Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In *P. J. Lee (Ed.): Advances in Cryptology-Proceedings of Asiacrypt 2004, LNCS 3329*, pages 48-62, Springer-Verlag, 2004.

[9] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *D. Stinson (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1993, LNCS 773*, pages 273-289, Springer-Verlag, 1993.

[10] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In*ACM Conference on Computer and Communications Security*, pages 62-73, 1993.

[11] E. Bresson and M. Manulis. Securing Group Key Exchange Against Strong Corruptions. AsiaCCS'08, pages 249-260.

[12] R. Canetti. Security and Composition of Cryptographic Protocols: A Tutorial. SIGACT News, 37(3,4), 2006.

[13] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. STOC 1998, pages 209-218, ACM.

[14] R. Canetti, O. Goldreich and S. Halevi. On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes. In 1st Theory of Cryptography Conference (TCC), LNCS 2951 , pages 40-57, Springer-Verlag, 2004.

[15] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, Springer-Verlag, 2001. Available also from Cryptology ePrint Archive, Report No. 2001/040.

[16] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In *M. Yung (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 143-161, Springer-Verlag, 2002.

[17] R. Cramer. Modular Design of Secure, yet Practical Cryptographic Protocols, PhD Thesis, University of Amsterdam, 1996.

[18] C. Cremers. Examining IndistinguishabilityCBased Security Models for Key Exchange Protocols: The case of CK, CKCHMQV, and eCK. In AsiaCCS 2011. Preliminary version available from Cryptology ePrint Archive, Report 2009/253.

[19] I. Damgård. Towards Practical Public-Key Systems Secure Against Chosen Ciphertext Attacks. In *J. Feigenbaum (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1991, LNCS 576*, pages 445-456. Springer-Verlag, 1991.

[20] A. Dent. Cramer-Shoup Encryption Scheme is Plantext Aware in the Standard Model. In *Advances in Cryptology-Proceedings of EUROCRYPT 2006, LNCS 4004*, pages 289-307. Springer-Verlag, 2006.

[21] M. Di Raimondo and R. Gennaro. New Approaches for Deniable Authentication. In proc. of 12nd ACM Conference on Computer and Communications Security (ACM CCS'05), ACM Press, pages 112-121, 2005.

[22] M. Di Raimondo, R. Gennaro and H. Krawczyk. Deniable Authentication and Key Exchange. ACM CCS'06, pages 466-475. Full version appears in Cryptology ePrint Archive Report No. 2006/280.

[23] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6): 644-654, 1976.

[24] V. S. Dimitrov, G. A. Jullien and W. C. Miller. Complexity and Fast Algorithms for Multiexponentiations. *IEEE Transactions on Computers*, 49(2): 141-147.

[25] Y. Dodis, J. Katz, A. Smith and S. Walfish. Composability and On-line Deniability of Authentication. Theory of Cryptography Conference (TCC), pages 146-162, 2009.

[26] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital Sigantures. In Crypto'89, pages 263-277.

[27] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *A. Odlyzko (Ed.): Advances in Cryptology-Proceedings of CRYPTO'86, LNCS 263*, pages 186-194. Springer-Verlag, 1986.

[28] FIPS Pub 186-2, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2, US Department of Commerce/National Institute of Standard and Technology, Githersburg, Maryland, USA, January 27, 2000. (Chance notice is made on October 5 2001.)

[29] J. A. Garay, P. D. MacKenzie, M. Prabhakaran and Ke Yang. Resource Fairness and Composability of Cryptographic Protocols. *Journal of Cryptology*, 24(4): 615-658 (2011).

[30] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All language in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery*, 38(1): 691-729, 1991.

[31] S. Goldwasser and Y. Lindell. Secure Computation without Agreement. *Journal of Cryptology*, 18(3), 247C287 (2005).

[32] S. Goldwasser, S. Micali and C. Rackoff. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIMA Journal on Computing*, 17(2): 281-308, 1988.

[33] D. M. Gordon. A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27(1): 129-146, 1998.

[34] L. Guillou and J. J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In *C. G. Gnther (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1988, LNCS 330* , pages 123-128, Springer-Verlag, 1988.

[35] S. Hada and T. Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *H. Krawczyk (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1998, LNCS 1462* , pages 408-423, Springer-Verlag, 1998.

[36] K. Hickman. The SSL Protocol. Online document, Feburary 1995. Available at www.netscape.com/eng/security/SSL-2.html.

[37] IEEE 1363-2000: Standard Specifications for Public Key Cryptography.

[38] ISO/IEC IS 15946-3. Information Technology - Security Techniques - Cryptographic Techniques Based on Elliptic Curves - Part 3: Key Establishment, 2002.

[39] B. Kaliski. An Unknown Key-Share Attack on the MQV Key Agreement Protocol. *ACM Transactions on Information and System Security (TISSEC)*, 4(3): 275-288, 2001.

[40] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. The Internet Engineering Task Force: INTERNET-DRAFT, October 2002.

[41] H. Krawczyk. SIGMA: the "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. Invited Talk at *D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 400-425, Springer-Verlag, 2003.

[42] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *V. Shoup (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2005, LNCS 3621*, pages 546-566. Springer-Verlag, 2005.

[43] H. Krawczyk. HMQV in IEEE P1363. July 2006.

[44] S. Kunz-Jacques and D. Pointcheval. A New Key Exchange Protocol Based on MQV Assuming Public Computations. In SCN'06, LNCS 4116, pages 186-200, Springer-Verlag, 2006.

[45] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Designs, Codes and Cryptography*, 28: 119-134, 2003.

[46] W. Mao. Modern Cryptography: Theory and Practice. Prentice Hall PTR, 2004.

[47] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems, Trans. IECE of Japan, 1986, E69(2), pp. 99-106.

[48] U. Maurer and S. Wolf. Diffie-Hellman Oracles. In *Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, pages 268-282, Springer-Verlag, 1996.

[49] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1995, pages 617-619.

[50] A. Menezes, M. Qu, and S. Vanstone. Some New Key Agreement Protocols Providing Mutual Implicit Authentication. Second Workshop on Selected Areas in Cryptography (SAC'95), 1995.

[51] A. Menezes and B. Ustaoglu. On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols. *INDOCRYPT* 2006: 133-147.

[52] C. J. Mitchell, M. Ward and P. Wilson. Key Control in Key Agreement Protocols. Electronic Letters, 34(10):980-981, 1998.

[53] M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 1(2): 231-262 (2004).

[54] D. Naccache, D. M'Raihi, S. Vaudenay and D. Raphaeli. Can D.S.A be Improved? Complexity Trade-Offs with the Digital Signature Standard. In *Advances in Cryptology-Proceedings of EUROCRYPT 1994, LNCS 950*, pages 77-85, Springer-Verlag, 1994.

[55] J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. In *Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 111-126, Springer-Verlag, 2002.

[56] NIST Special Publication 800-56 (DRAFT): Recommendation on Key Establishment Schemes. Draft 2, January 2003.

[57] NSAs Elliptic Curve Licensing Agreement. Presentation by Mr. John Stasak (Cryptography Office, National Security Agency) to the IETF's Security Area Advisory Group, November 2004.

[58] T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In PKC'01, LNCS 1992, pages 104-118, Springer-Verlag, 2001.

[59] R. Pass. On Deniabililty in the Common Reference String and Random Oracle Models. In *Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 316-337, Springer-Verlag 2003.

[60] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13: 361-396, 2000.

[61] C. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3): 161-174, 1991.

[62] A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In In *Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 355-367, Springer-Verlag, 1996.

[63] SP 800-56 (DRAFT), Special Publication 800-56, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology, July 2005.

[64] T. Ylonen. SSH Protocol Architecture. INTERNET-DRAFT, draft-ietf-architecture-15.txt, 2002.

[65] T. Ylonen. SSH Transport Layer Protocol. INTERNET-DRAFT, draft-ietf-architecture-13.txt, 2002.

[66] A.C.Yao and Y. Zhao. On-line Efficient, Deniable and Non-Malleable Key-Exchange Methods (in Chinese), Domestic patent in China, No. 200710047344.8, August 2007.

[67] A.C.Yao and Y. Zhao. Method and Structure for Self-Sealed Joint Proof-of-Knowledge and Diffie-Hellman Key-Exchange Protocols. PCT Patent. Online available from Global Intellectual Property Office (GIPO) since August 2008. This is the PCT version of [66], with [66] serving as the priority reference.

[68] A.C.Yao and Y. Zhao. Deniable Internet Key-Exchange. In *8th International Conference on Applied Cryptography and Network Security (ACNS 2010), LNCS 6123* , pages 329-348, Springer-Verlag, 2010. Preliminary version appears in Cryptology ePrint Archive, Report 2007/191.

A Variants of (H)MQV

Three-round HMQV (resp., MQV) adds key confirmation as follows: let $K_m = H_K(K_{\hat{A}}, 0) = H_K(K_{\hat{B}}, 0)$, $\hat{B}$ uses K_m as the MAC key to authenticate 0 (resp., $(2, \hat{B}, \hat{A}, Y, X)$) in the second-round of HMQV (resp., MQV); and $\hat{A}$ uses K_m to authenticate 1 (resp., $(3, \hat{A}, \hat{B}, X, Y)$) in an additional third-round of HMQV (resp., MQV). The session-key is set to be $K = H_K(K_{\hat{A}}, 1) = H_K(K_{\hat{B}}, 1)$.

In one-round HMQV, only $\hat{A}$ sends X, and the session-key is derived as follows: $K_{\hat{A}} = B^{x+da}$, $K_{\hat{B}} = (XA^d)^b$, $d = h(X, \hat{A}, \hat{B})$, $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$.

B Basic Definitions

Gap Diffie-Hellman (GDH) assumption [58]. Let G be a cyclic group generated by an element g, and a decision predicate algorithm $\mathcal{O}$ be a (*full*) Decisional Diffie-Hellman (DDH) Oracle for the group G and generator g such that on input (U, V, Z), for *arbitrary* $(U, V) \in G^2$, oracle $\mathcal{O}$ outputs 1 if and only if $Z = CDH(U, V)$. We say the GDH assumption holds in G if for any polynomial-time CDH solver for G, the probability that on a pair of random elements $(X, Y) \leftarrow G$ the solver computes the correct value $CDH(X, Y)$ is negligible, even when the algorithm is provided with the (full) DDH-oracle $\mathcal{O}$ for G. The probability is taken over the random coins of the solver, and the choice of X, Y (each one of them is taken uniformly at random in G).

Knowledge-of-Exponent Assumption (KEA). Let G be a cyclic group of prime order q generated by an element g, and consider algorithms that on input a triple $(g, C = g^c, z)$ output a pair $(Y, Z) \in G^2$, where c is taken uniformly at random from Z_q^* and $z \in \{0, 1\}^*$ *is an arbitrary string that is generated independently of* C. Such an algorithm $\mathcal{A}$ is said to be a KEA algorithm if with non-negligible probability (over the choice of g, c and $\mathcal{A}$'s random coins) $\mathcal{A}(g, g^c, z)$ outputs $(Y, Z) \in G^2$ such that $Z = Y^c$. Here, $C = g^c$ is the random challenge to the KEA algorithm $\mathcal{A}$, and z captures the auxiliary input of $\mathcal{A}$ that is independent of the challenge C.

We say that the KEA assumption holds over G, if for every probabilistic polynomial-time (PPT) KEA algorithm $\mathcal{A}$ for G there exists another efficient algorithm $\mathcal{K}$, referred to as the KEA-extractor, for which the following property holds except for a negligible probability: let (g, g^c, z) be an input to $\mathcal{A}$ and ρ a vector of random coins for $\mathcal{A}$ on which $\mathcal{A}$ outputs $(Y, Z = Y^c)$, then, on the same inputs and random coins, $\mathcal{K}(g, C, z, \rho)$ outputs the triple $(Y, Z = Y^c, y)$ where $Y = g^y$.

C NMJPOK: Motivation, Formulation, and Implementations

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a Diffie-Hellman protocol $\langle \hat{A}, \hat{B} \rangle$ are run concurrently over an asynchronous network like the Internet. To distinguish concurrent sessions, each session run at the side of an uncorrupted player is labeled by a tag, which is the concatenation, in the order of session initiator and then session responder, of players' identities/public-keys and DH-components available from the session transcript. A session-tag is complete if it consists of a complete set of all these components.

In this work, we study the mechanisms for *non-malleably* and *jointly* proving the knowledge of both b and y w.r.t. a challenge DH-component X between the prover $\hat{B}$ (of public-key $B = g^b$ and DH-component $Y = g^y$) and the verifier $\hat{A}$ (who presents the challenge DH-component $X = g^x$), where $b, y, x \leftarrow Z_q^*$. Our starting point is the JPOK mechanism proposed in [68] for deniable Internet key-exchange: $JPOK_{(b,y)} = h(\hat{A}, A, \hat{B}, B, Y, X, X^b, X^y)$ w.r.t. a random DH-component challenge X from $\hat{A}$. This JPOK mechanism is shown to be sound in the RO model [68], but is less efficient and has no way to be used for IA-OAKE. Motivated to further improve its efficiency toward a building tool for IA-OAKE, in this work we investigate joint proof-of-knowledge (JPOK) of the type $JPOK_{(b,y)} = f_0^h(X^b, aux_0) \cdot f_1^h(X^y, aux_1)$ in the random oracle model, where f_0^h and f_1^h are some functions from $\{0,1\}^*$ to $G \setminus 1_G$ with oracle access to an RO $h : \{0,1\}^* \rightarrow \{0,1\}^l$, aux_0 and aux_1 are some public values. Moreover, we look for solutions of $JPOK_{(b,y)}$ such that $JPOK_{(b,y)}$ can be efficiently computed with one single exponentiation by the knowledge prover. Note that the tag for a complete session of $JPOK_{(b,y)}$ is $(\hat{A}, \hat{B}, B, X, Y)$. The possibility of NMJPOK without ROs (based upon pairings) is left to be studied in a subsequent separate paper. In the rest of this paper, we denote by the output length, i.e., l, of h as the security parameter.

One naive solution of $JPOK_{(b,y)}$ is just to set $JPOK_{(b,y)} = X^b \cdot X^y = X^{b+y}$. But, such a naive solution is totally insecure, for example, an adversary $\mathcal{A}$ can easily impersonate the prover $\hat{B}$ and pre-determine the value of $JPOK_{(b,y)}$ to be 1_G, by simply setting $Y = B^{-1}$. The underlying reason is: $\mathcal{A}$ can malleate B and Y into X^{y+b} *by maliciously correlating the exponents of y and b*, but actually without knowing either of them. A further remedy of this situation is to mask the exponents b and y by some random values. In this case, the proof is denoted as $JPOK_{(b,y)} = X^{db+ey}$, where d and e are random values (e.g., $d = h(X, \hat{B})$ and $e = h(Y, \hat{A})$). The intuition with this remedy solution is: since d and e are random values, the values of db and ey are also random (even if the values Y and B, and thus the values of y and b, may be maliciously correlated). This intuition however turns out also to be wrong *in general*. With the values $d = h(B, \hat{A})$ and $e = h(X, \hat{B})$ as an illustrative example, after receiving X an adversary $\mathcal{A}$ can generate and send $Y = B^{-d/e}$, and in this case $JPOK_{(b,y)} = X^{db+ey} = 1_G$. This shows that masking b and y by random values is also not sufficient for ensuring the non-malleability of $JPOK_{(b,y)}$. The key point here is that the values db and ey are *not* necessarily *independent*. A series of careful investigations bring us to the following principles for proving DH knowledges non-malleably and jointly:

Inside Computational Independence. Denote $S_0 = \{X, B\}$, $Z_0 = CDH(X, B) = g^{xb}$, $F_0 = f_0^h(Z_0, aux_0)$, $S_1 = \{X, Y\}$, $Z_1 = CDH(X, Y)$ and $F_1 = f_1^h(Z_1, aux_1)$. The key principle is: the inside multiplied components F_0 and F_1 of $JPOK_{(b,y)}$ should be *computationally independent*, no matter how a malicious knowledge prover $\hat{B}$ (of public-key $B = g^b \in G$) does. That is, the adversarial attempts at Z_δ for any $\delta \in \{0,1\}$ should be essentially *sealed* (i.e., localized) to F_δ, and are isolated (i.e., "independent") from the adversarial attempts at $Z_{1-\delta}$. This essentially ensures that no matter how the possibly malicious knowledge-prover $\hat{B}$ does, to compute $JPOK_{(b,y)}$ $\hat{B}$ has to compute two "*independent*" DH-secrets F_0 and F_1 w.r.t. the fresh challenge X, which implies that $\hat{B}$ does indeed "know" both b and y.

Definition C.1 (computational independence) *We formulate two types of "computational independence" w.r.t. $JPOK_{(b,y)}$:*

(1) Self-sealed computational independence. Given arbitrary values $(\alpha, \beta) \in (G \setminus 1_G)^2$, no matter how a malicious $\hat{B}$ does, both $\Pr[F_0 = \alpha]$ *and* $\Pr[F_1 = \beta]$ *are negligible.*

(2) Committed computational independence. There exists $\delta \in \{0,1\}$ such that for any $\alpha \in G \setminus 1_G$ $\Pr[F_\delta = \alpha]$ is negligible, no matter how a malicious $\hat{B}$ does. This captures the independence of F_δ on $F_{1-\delta}$, i.e., the infeasibility of adversarial attempts by a malicious prover on setting F_δ to be correlated to $F_{1-\delta}$; On the other hand, the value $F_{1-\delta}$ is committed to F_δ, in the sense that

- $S_{1-\delta} \bigcup aux_{1-\delta} \subseteq aux_\delta$.
- *Given (Z_δ, aux_δ) that determines $F_\delta = f_\delta^h(Z_\delta, aux_\delta)$, no efficient algorithm can provide, with non-negligible probability, $(S'_{1-\delta}, aux'_{1-\delta}) \subseteq aux'_\delta$ (w.r.t. the same challenge $X = S_{1-\delta} \cap S'_{1-\delta}$ from $\hat{A}$ and $aux_\delta - aux_{1-\delta} = aux'_\delta - aux'_{1-\delta}$) such that $S'_{1-\delta} \bigcup aux'_{1-\delta} \neq S_{1-\delta} \bigcup aux_{1-\delta}$ but $f_\delta^h(Z_\delta, aux_\delta) = f_\delta^h(Z_\delta, aux'_\delta)$.That is, any adversarial attempt by a malicious prover on setting $F_{1-\delta}$ to be correlated to a given value F_δ, by changing $\{S_{1-\delta}, aux_{1-\delta}\}$ into $\{S'_{1-\delta}, aux'_{1-\delta}\}$ w.r.t. the same random challenge $X = S_{1-\delta} \cap S'_{1-\delta}$ and $aux_\delta - aux_{1-\delta} = aux'_\delta - aux'_{1-\delta}$ (for example, by simply changing B for the case of $\delta = 1$ or Y for the case of $\delta = 0$), will cause the value F_δ itself changed that in turn determines and commits to the value $F_{1-\delta}$ (while $\Pr[F_\delta = \alpha]$ is negligible for any $\alpha \in G \setminus 1_G$). This implies the infeasibility of adversarial attempt on setting $F_{1-\delta}$ to be correlated to F_δ, i.e., the "computational independence" of $F_{1-\delta}$ on F_δ.*

The probabilities are taken over the random coins used by the malicious $\hat{B}$ and the honest $\hat{A}$, and the choice of the random function h in the RO model.

Informally speaking, the underlying rationale of $NMJPOK_{(b,y)}$ is: given a random challenge X, no matter how a malicious $\hat{B}$ chooses the values $Y = g^y$ and $B = g^b$ (where the values y and b can be arbitrarily correlated), it actually has no control over the values db and ey in the RO model. Specifically, for each given complete session tag and arbitrary $(\alpha, \beta) \in (G \setminus 1_G)^2$ the probability of $F_0 = \alpha$ or $F_1 = \beta$ is at most $\frac{2}{2^l-1}$, and suppose the malicious prover works in $T = poly(l)$ time, then by the birthday paradox the probability that there exist a session in which $F_0 = \alpha$ or $F_1 = \beta$ is at most $\frac{2T^2}{2^l-1}$, which is still negligible in l. Alternatively speaking, given a random challenge X, (by the birthday paradox) it is infeasible for a malicious $\hat{B}$ to output $B = g^b$ and $Y = g^y$ such that the values db and ey satisfy some predetermined (polynomial-time computable) relation with non-negligible probability in the RO model.

The situation with $sNMJPOK_{(b,y)}$ is a bit different. Though as in $NMJPOK_{(b,y)}$, the malicious prover $\hat{B}$ is infeasible to set ey to a predetermined value, $\hat{B}$ can always set the value $db = b$ at its wish as $d = 1$ for $sNMJPOK_{(b,y)}$. But, $\hat{B}$ is still infeasible to set the value b correlated to $ey = h(B, X, Y)y$, particularly because the value B is put into the input of (i.e., commits to) e. Specifically, for any value B (that determines the value b) set by $\hat{B}$, with the goal of making b and ey correlated, the probability that the values $ey = h(B, X, Y)y$ and b satisfy some predetermined (polynomial-time computable) relation is negligible in the RO model (again by the birthday paradox). In particular, by the birthday paradox, the probability that $\Pr[b = f(ey)]$ or $\Pr[f(b) = ey]$, where f is some predetermined polynomial-time computable function (that is in turn determined by some predetermined polynomial-time computable relation), is negligible in the RO model, no matter how the malicious $\hat{B}$ does.

Outside Non-Malleability. As JPOK may be composed with other protocols in practice, another principle is that the JPOK provided by one party in a session should be bounded to that session, in the sense that the JPOK should not be malleated *into* or *from* other sessions. This is captured by the following definition, which particularly implies the property of "key control" [45] for DHKE.

Definition C.2 (tag-binding self-seal (TBSS)) *For a DH protocol in the RO model, denote by Z_{Tag} the random variable of the shared DH-secret in G (say, JPOK or session-key) determined by a complete session-tag Tag (taken over the choice of the random function h in the RO model). We say it is* tag-binding self-sealed, *if for any $\alpha \in G \setminus 1_G$ and any complete Tag, $\Pr[Z_{Tag} = \alpha] \leq O(\frac{1}{2^l})$ where l is the security parameter. The probability is taken over the choice of the random function h in the RO model.*

The definition of TBSS particularly implies that: given an arbitrary yet complete session-tag Tag, by the birthday paradox no efficient (polynomial-time) algorithm can, *with non-negligible probability*, output a different $Tag' \neq Tag$ such that $Z_{Tag'}$ and Z_{Tag} collide in the sense $Z_{Tag'} = Z_{Tag}$ in the RO model assuming h is a random function. In more detail, by the birthday paradox, the probability that an efficient algorithm finds two colliding tags (Tag, Tag') such that $Z_{Tag} = Z_{Tag'}$ is bounded by $O(\frac{T^2}{2^l})$, where $T = poly(l)$ is the running time of the algorithm. In a sense, the DH-secret determined by a complete session-tag is "bounded" to this specific session, and is essentially "independent" of the outside world composed concurrently with the current session. In particular, the shared DH-secret is random and unpredictable.

TBSS vs. contributiveness. The work [2] introduced the notion of "contributiveness" property for password-authenticated group key exchange protocols, which roughly says that the distributions of session-keys are guaranteed to be random, as long as there are enough honest players in a session. We noted that our TBSS definition, originally presented in [66, 67] independently of [2], has similar security guarantee. As we shall see, (H)MQV lacks the TBSS property by the EDA attacks presented in Section 5, which implies also that the TBSS property is not captured by the CK-framework.

We say that $JPOK_{(b,y)}$ is a non-malleable joint proof-of-knowledge (NMJPOK), of the knowledges (b, y) w.r.t. the random DH-component challenge X, if $JPOK_{(b,y)}$ satisfies both the above two principles.

Remark: The TBSS property simplifies the analysis of (s,r)OAKE in the CK-framework. The computational independence property is not directly used in the analysis in the CK-framework, but it (together with the TBSS property) is fundamental to security beyond the CK-framework as discussed in detail in Section 5 and Appendix H.

Preferable candidates for NMJPOK. Guided by the above principles, we propose two preferable solutions for NMJPOK in the RO model:

- Self-sealed JPOK (SSJPOK): $SSJPOK_{(b,y)} = X^{db+ey}$, where $d = h(\hat{A}, \hat{B}, B, X)$ and $e = h(X, Y)$; Specifically, $aux_0 = \{\hat{A}, \hat{B}, B, X\}$ and $aux_1 = \{X, Y\}$, $F_0 = f_0^h(X^b, aux_0) = X^{bh(aux_0)}$ and $F_1 = f_1^h(X^y, aux_1) = X^{yh(aux_1)}$. Here, $h : \{0,1\}^* \to \{0,1\}^l/0 \subseteq Z_q^*$ is a hash function and $l \approx |q|$ (in the unlikely case that $h(x) = 0$ for some x, the output of $h(x)$ can be defined by default to be a value in $Z_q^* \setminus \{0,1\}^l$).
- Single-hash SSJPOK (sSSJPOK): $sSSJPOK_{(b,y)} = X^{db+ey}$, where $d = 1$ and $e = h(\hat{A}, \hat{B}, B, X, Y)$; Specifically, aux_0 is empty and $aux_1 = \{\hat{A}, \hat{B}, B, X, Y\}$, $F_0 = f_0^h(X^b, aux_0) = X^b$ and $F_1 = f_1^h(X^y, aux_1) = X^{yh(aux_1)}$.

Needless to say, there may possibly be other NMJPOK candidates, but the above explicitly proposed solutions enjoy the following advantageous properties, which make them more desirable:

- Post-ID, modular and offline computability of SSJPOK. Specifically, as the input of e does not include $\hat{A}$'s identity and public-key, $\hat{A}$ can first send X without revealing its identity information. In this case, $\hat{B}$ can first compute X^{ey}, and then X^{db} only after learning $\hat{A}$'s identity and public-key. Also, without inputting Y into d allows $\hat{A}$ to pre-compute $B^{dx} (= X^{db})$ prior to the protocol run.
- sSSJPOK is preferable because of its offline computability, more efficient computational complexity and the less use of hash function h.

It is quite straightforward to check that, in the RO model, SSJPOK (resp., sSSJPOK) satisfies self-sealed (resp., committed) computational independence, and both of them are tag-binding self-sealed. In more details, for SSJPOK, for any given values (B, Y) (which determine (b, y)) output by a malicious prover $\hat{B}$ and any value $\hat{\beta} \in Z_q^*$ $\Pr[db = \hat{\beta}]$ (resp., $\Pr[ey = \hat{\beta}]$) is constant: either 0 or $\frac{1}{2^l-1}$ in the RO model (no matter how a malicious prover $\hat{B}$ does). The committed computational independence of sSSJPOK is from the observation: $\{X, B\}$ (that determines $F_0 = X^b$) are committed to $F_1 = X^{yh(aux_1)}$ in the RO model as $\{X, B\} \subseteq aux_1$. The TBSS property of (s)SSJPOK can be derived by a straightforward calculation. Proof details that (s)SSJPOK are NMJPOK in the RO model are given below.

Proposition C.1 *SSJPOK is NMJPOK in the RO Model.*

Proof. We first prove the self-sealed computational independence of SSJPOK in the RO model. Note that for SSJPOK, $F_0 = X^{db} = X^{h(\hat{A},\hat{B},B,Y)b}$ and $F_1 = X^{ey} = X^{h(X,Y)y}$, where $b, y, x \leftarrow Z_q^*$. For any given challenge $X \in G \setminus 1_G$, each pair of values $(B = g^b, Y = g^y) \in (G \setminus 1_G)^2$ (that determine

$(b, y) \in (Z_q^*)^2)$ and any pair of given values $\alpha = g^{\hat{\alpha}}, \beta = g^{\hat{\beta}} \in (G \setminus 1_G)^2$, where $\hat{\alpha}, \hat{\beta} \in Z_q^*$, we consider the set of values that F_0 can be assigned in the RO model $S_{F_0} = \{X^{db} | 0 \leq d = h(\hat{A}, \hat{B}, B, Y) \leq 2^l - 1\}$ and also the set of values that F_1 can be assigned in the RO model $S_{F_1} = \{X^{ey} | 0 \leq e = h(X, Y) \leq 2^l - 1\}$. If $\alpha \notin S_{F_0}$ or $d = 0$ (resp., $\beta \notin S_{F_1}$ or $e = 0$), then we have $\Pr[F_0 = \alpha] = 0$ (resp., $\Pr[F_1 = \beta] = 0$). If $\alpha \in S_{F_0}$ (resp., $\beta \in S_{F_1}$), then we have $\Pr[F_0 = \alpha] = \frac{1}{2^l - 1}$ (resp., $\Pr[F_1 = \beta] = \frac{1}{2^l - 1}$) in the RO model. As the malicious prover $\hat{B}$ is polynomial-time, we have that, no matter the polynomial-time malicious $\hat{B}$ does on a challenge X, the probability that it outputs B, Y such that $F_0 = \alpha$ and $F_1 = \beta$ is negligible. Specifically, suppose $N = 2^l - 1$ and $T = poly(l)$ is the running time of $\hat{B}$, by the birthday paradox the probability that on input (X, α, β) the malicious $\hat{B}$ outputs (B, Y) such that $F_0 = \alpha$ or $F_1 = \beta$ is at most $\frac{T(T-1)}{2N}$ that is negligible (in l).

Next we prove the TBSS property of SSJPOK in the RO model, which is based on and can be easily derived from the NMJPOK property of OAKE. For a complete session of $SSJPOK$, its tag is: $Tag = (\hat{A}, \hat{B}, B = g^b, X = g^x, Y = g^y)$, where $b, x, y \in Z_q^*$, we consider the value $Z_{Tag} = X^{db+ey} = X^{h(\hat{A}, \hat{B}, B, Y)b} \cdot X^{h(X,Y)y}$ in the RO model where h is assumed to be a random oracle. As for each value $\alpha \in G \setminus 1_G$, $\Pr[X^{h(\hat{A}, \hat{B}, B, Y)b} = \alpha] \leq \frac{1}{2^l - 1}$ and $\Pr[X^{h(X,Y)y} = \alpha] \leq \frac{1}{2^l - 1}$ in the RO model, we get (by straightforward calculation) that $\Pr[Z_{Tag} = \alpha] \leq O(\frac{1}{2^l})$. □

Proposition C.2 *sSSJPOK is NMJPOK in the RO model.*

Proof. We first show the committed computational independence property of sSSJPOK. Similar to the analysis of Proposition C.1, for the case $\delta = 1$ we have that for any given $\alpha \in G \setminus 1_G$ and any DH-component challenge X, and any $(B, Y) \in (G \setminus 1_G)^2$, $\Pr[F_\delta = X^{yh(aux_1)} = X^{yh(\hat{A}, \hat{B}, B, X, Y)} = \alpha] \leq \frac{1}{2^l - 1}$ in the RO model, where $\delta = 1$. As the malicious $\hat{B}$ is polynomial-time, we have the probability that the malicious $\hat{B}$ outputs (B, Y), given a random challenge X and a given value $\alpha \in G \setminus 1_G$, such that $F_1 = \alpha$ is negligible in the RO model.[4] Then, the committed computational independence of sSSJPOK is from the following observation that X^b is committed to $X^{yh(\hat{A}, \hat{B}, B, X, Y)}$. Specifically,

- $S_{1-\delta} = S_0 = \{X, B\} \subseteq aux_\delta = aux_1 = \{\hat{A}, \hat{B}, B, X, Y\}$. Note that the value $F_0 = Z_0 = X^b$ (resp., $F_1 = f_1^h(Z_1, aux_1) = f_1^h(X^y, aux_1) = X^{yh(aux_1)} = X^{yh(\hat{A}, \hat{B}, B, X, Y)}$) is determined by $S_0 = \{X, B\}$ (resp., $aux_1 = \{\hat{A}, \hat{B}, B, X, Y\}$), and aux_0 is empty for sSSJPOK.
- Given $Z_\delta = Z_1 = X^y$ and $aux_\delta = aux_1 = \{\hat{A}, \hat{B}, B, X, Y\}$, for any $B' \neq B$ such that $S'_0 = \{X, B'\} \subseteq aux'_1 = \{\hat{A}, \hat{B}, B', X, Y\}$, we get $\Pr[f_1^h(Z_1, aux_1) = f_1^h(Z_1, aux'_1)] = \Pr[X^{yh(\hat{A}, \hat{B}, B, X, Y)} = X^{yh(\hat{A}, \hat{B}, B', X, Y)}] \leq \frac{1}{2^l - 1}$. Thus for any polynomial-time algorithm, the probability that it, on input Z_1, aux_1, outputs $S'_0 = \{X, B'\}$ for $B' \neq B$ such that $X^{yh(\hat{A}, \hat{B}, B, X, Y)} = X^{yh(\hat{A}, \hat{B}, B', X, Y)}$ is negligible (again by the birthday paradox).

Next, we show the TBSS property of sSSJPOK in the RO model, which is based on and can be easily derived from the NMJPOK property of sSSJPOK. For the tag $Tag = (\hat{A}, \hat{B}, B, X, Y)$ of a complete session of sSSJPOK, we consider the value $Z_{Tag} = X^{b + yh(\hat{A}, \hat{B}, B, X, Y)} = X^b \cdot X^{yh(\hat{A}, \hat{B}, B, X, Y)}$. No matter what value X^b is, for any value $\alpha \in G \setminus 1_G$ we have $\Pr[X^{yh(\hat{A}, \hat{B}, B, X, Y)} = \alpha] \leq \frac{1}{2^l - 1}$ in the RO model. Thus, for any value $\alpha \in G \setminus 1_G$ we have also that $\Pr[Z_{Tag} = \alpha] \leq \frac{1}{2^l - 1} = O(\frac{1}{2^l})$. □

D More Variants of (s,r)OAKE

One-round OAKE (oOAKE): The player $\hat{A}$ sends $X = g^x$ to $\hat{B}$. Normally, $\hat{A}$ is a client machine and $\hat{B}$ is a server machine. Let $K_{\hat{A}} = B^{a+ex}$ and $K_{\hat{B}} = A^b X^{eb}$, where $e = h(\hat{A}, A, \hat{B}, B, X)$ and the session-key is $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$. For oOAKE, it is also recommend to set the output length

[4] Specifically, by the birthday paradox, the probability is at most $O(\frac{T^2}{2^l})$, where $T = poly(l)$ is the running time of $\hat{B}$.

of h to be shorter, e.g., $|q|/2$, to ease the computation of $K_{\hat{B}} = A^b X^{eb} = (AX^e)^b$ in some application scenarios (e.g., when the pre-computation of A^b is inconvenient).

Note that the computational complexity of $\hat{A}$ is 2 exponentiations in total and all the computation of $\hat{A}$ can be offline. To improve the on-line efficiency of $\hat{B}$, the player $\hat{B}$ can pre-compute A^b in an off-line way (and store it in a database entry corresponding to the client $\hat{A}$), and only on-line computes X^{eb} and X^q which amounts to about 1.2 exponentiations (it is recommended for $\hat{B}$ to explicitly check the subgroup membership of X). In case of embedded subgroup test, $\hat{B}$ should explicitly check $X \in G'$ and $X^{ebt} \neq 1_G$ (only checking $K_{\hat{B}} \neq 1_G$ is not sufficient to prevent the small subgroup attack). We remind that oOAKE intrinsically suffers from the key compromise impersonation (KCI) vulnerability in case $\hat{B}$'s static secret-key b is compromised, and lacks perfect forward secrecy (the same vulnerabilities hold also for one-round variant of HMQV).

Adding (explicit) mutual authentication. For adding mutual authentication to (s,r)OAKE, besides the session-key K we also need a MAC-key K_m to be used within the protocol run (but erased after the protocol run). Both the session-key and MAC-key are derived from the shared DH-secret $K_{\hat{A}} = K_{\hat{B}}$, and are independent in the random oracle model. For (s,r)OAKE with mutual authentication, $\hat{B}$ sends an additional value $t_B = MAC_{K_m}(1)$ in the second-round, and $\hat{A}$ sends $t_A = MAC_{K_m}(0)$ in an additional third-round. For one-round OAKE with mutual authentication, the player $\hat{A}$ can additionally send $t_A = MAC_{K_m}(0)$ in the first-round, and the player $\hat{B}$ responds back $MAC_{K_m}(1)$ in the subsequent round. In practice, the message authentication code MAC can be instantiated with HMAC [6].

E More Discussions on the Specification of (s,r)OAKE

Subgroup test vs. ephemeral DH-exponent leakage. We note that the damage caused by ignoring the subgroup test of peer's DH-component (but still with the supergroup G' membership check) can be much relieved (and even waived), if the ephemeral private values generated within the protocol run are well-protected. For example, even if an adversary learns some partial information about $db + ey$ by issuing a small subgroup attack against the honest $\hat{B}$ (by setting X to be in a small subgroup), it still cannot derive the value b without compromising the ephemeral value y. Also note that the adversary actually cannot derive the full value of $db + ey$ by small subgroup attacks, as the DH-exponent y is independent at random in each session. In this case, we suggest that embedded subgroup test is sufficient. For presentation simplicity and unity, throughout this paper, it is assumed that $t = \frac{N}{q}$ for implementations with embedded subgroup test, and $t = 1$ with explicit subgroup test.

Ephemeral private values exposable to adversary. The ephemeral values exposable to adversary, generated by the honest $\hat{B}$ (resp., $\hat{A}$) during the protocol run, are specified to be: (y, Y) (resp., (x, X)) if $\hat{B}$ (resp., $\hat{A}$) does not pre-compute A^{bf+cy} (resp., B^{af+dx}), where $f = 0$ for (s)OAKE and $f = 1$ for rOAKE. But, for (s,r)OAKE with offline pre-computation of A^{fb+cy} and B^{fa+dx}, the values allowed to be exposed are different between (s)OAKE and rOAKE. For (s)OAKE, all the values (y, Y, A^{cy}) (resp., (x, X, B^{dx})) can be exposed. But, for rOAKE, the pre-computed value A^{b+cy} or B^{a+dx} is not allowed to be exposed for provable security in the CK-framework. More detailed clarifications are referred to Appendix G.1.2. Other ephemeral private values, other than (y, Y, A^{fb+cy}) and (x, X, B^{fa+dx}), are erased promptly after use. We remark all ephemeral private values, except for the session-key in case the session is successfully finished, generated by an honest party within the protocol run are erased after the session is completed (whether finished or aborted). For expired sessions, the session-keys are also erased.

F More Discussions on the Security of (s,r)OAKE vs. HMQV

Assuming *all* the DH-components generated by *all* uncorrupted players are *not* exposed to the attacker prior to the sessions involving them (e.g., all honest players only generate fresh ephemeral DH-components *on the fly*, i.e., without pre-computation, in each session), and assuming *all* the ephemeral DH-exponents generated during session runs are unexposed to the attacker, the SK-security of HMQV can be based on the CDH assumption, while we do not know how to prove this property with (s,r)OAKE. This is the only advantage of HMQV over (s,r)OAKE that we can see.

However, as already stressed in [42], security against exposed DH-exponents is deemed to be the main and prime concern for any robust DHKE, and security against exposed offline pre-computed values (particularly, the DH-components) is important to both lower-power devices and to high volume servers [42]. The reason is, as pointed out in [42], many applications in practice will boost protocol performance by pre-computing and storing values for later use in the protocol. In this case, however, these stored values are more vulnerable to leakage, particularly when DHKE is deployed in hostile environments plagued with spyware or virus and in view of that the offline pre-computed DH-components are much less protected in practice as they are actually public values to be exchanged in plain.

Also, for DHKE protocols running concurrently in settings like the Internet, we suggest it is unreasonable or unrealistic to assume non-precomputation and non-exposure of the *public* DH-components for *all* uncorrupted parties in the system. Note that, whenever there is an uncorrupted player whose DH-component is exposed prior to the session in which the DH-component is to be used (the attacker can just set this session as the test-session), the security of HMQV relies on both the GDH assumption and the KEA assumption in most cases as clarified in Appendix G.

For the above reasons, we suggest that the security advantage of HMQV over (s,r)OAKE in this special case is insignificant in reality. Note that, even in this special case, (s,r)OAKE enjoys other security advantages: (1) More robust embedded subgroup test supported by offline pre-computability of A^{fb+cy} and B^{fa+dx}. (2) Resistance to more powerful secrecy exposure of the additional pre-computed private values A^{cy} and B^{dx} for OAKE and sOAKE; (3) Stronger resistance against collision attacks on the underlying hash function h. Further note that, in the case of pre-computed and exposed DH-components, (s)OAKE is based upon weaker assumptions (i.e., only the GDH assumption) than (H)MQV (that is based on both the GDH assumption and the KEA assumption) for the most often case of $\hat{A} \neq \hat{B}$, and rOAKE totally gets rid of the non-standard KEA assumption.

G Security Analysis of (s,r)OAKE in the CK-Framework

One of main conceptual contributions of the analysis of HMQV in the CK-framework [42] is to cast the design of HMQV in terms of Hashed Dual challenge-Response (HDR) signatures and Hashed Challenge-Response (HCR) signatures, which are in turn based on Dual Challenge-Response (DCR) signatures and eXponential Challenge-Response (XCR) signatures and can be traced back to Schnorr's identification scheme [61]. We show that (s,r)OAKE all can be casted in terms of HDR signatures. Moreover, the HDR signatures implied by the (s,r)OAKE protocols, referred to as OAKE-HDR and sOAKE-HDR, are both *online efficient* and *strongly secure*. This provides extra security strength of the underlying building tools, say SSJOPK and sSSJPOK, used in (s,r)OAKE.

G.1 Casting (s,r)OAKE in Terms of *Online Efficient* and *Strongly Secure* HDR Signatures

Informally speaking, a HDR signature scheme is an *interactive* signature scheme between two parties in the public-key model. The two parties generate the *same* signature, which is actually a *hashed* value of the DH-secret shared between the two parties, with the *dual* roles of signer and challenger: each party generates the signature with private values of its static secret-key and the secret DH-exponent with respect to its peer's DH-component and public-key as the challenges. With a HDR signature, we are only interested to ensure verifiability of the signature by the two intended parties, and thus we make no assumptions or requirements regarding the transferability or verifiability of the signature by a third party. Roughly speaking, a HDR signature scheme is secure if the signature cannot be generated by any other parties other than the two intended (honest) parties.

Definition G.1 ((s,r)OAKE-HDR signature schemes) *Let $\hat{A}$, $\hat{B}$ be two parties with public-keys $A = g^a$, $B = g^b$, respectively. Let $m_{\hat{A}}$, $m_{\hat{B}}$ be two messages. The* (s,r)OAKE-HDR *signatures of $\hat{B}$ on* *messages $(m_{\hat{A}}, m_{\hat{B}}, \hat{A}, A, \hat{B}, B, X, Y)$, where $X = g^x$, $Y = g^y$ are chosen by $\hat{A}$, $\hat{B}$ respectively as the random* challenge *and* response, *$x, y \leftarrow Z_q^*$, $m_{\hat{A}}$ (resp., $m_{\hat{B}}$) includes arbitrary (maybe empty) messages sent to $\hat{A}$ (resp., $\hat{B}$) from $\hat{B}$ (resp., $\hat{A}$), are defined as a vector of values (the signatures of $\hat{A}$ are defined straightforwardly): $\{\hat{A}, A, m_{\hat{A}}, m_{\hat{B}}, X, Y, HSIG^{OAKE}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y) = H_K(A^{bf+yc}X^{bd+ye})\}$, where*

OAKE-HDR. *$f = 0$, $c = h(m_{\hat{A}}, \hat{A}, A, Y)$, $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$.*

sOAKE-HDR. *$f = 0$, $c = d = 1$ and $e = h(m_{\hat{A}}, m_{\hat{B}}, \hat{A}, A, \hat{B}, B, X, Y)$.*

rOAKE-HDR. *$f = 1$, $c = h(m_{\hat{A}}, \hat{A}, A, Y)$, $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$.*

Note that the online efficiency of (s,r)OAKE-HDR can be only one exponentiation for each player. In comparison, each player of HMQV-HDR performs about 1.3 online exponentiations. For presentation simplicity, in the above HDR signature description we assume the CA in the underlying PKI will check the membership $G \setminus 1_G$ of registered public-keys, and each player checks the membership $G \setminus 1_G$ of its peer's DH-component.

(s,r)OAKE in a nutshell. Actually, the above (s,r)OAKE-HDR can be viewed as a general structure of the (s,r)OAKE protocols. Specifically, (s,r)OAKE are instantiated with (s,r)OAKE-HDR respectively, with the special $m_{\hat{A}}$ and $m_{\hat{B}}$ set to be the empty string. In general, $m_{\hat{A}}$ (resp., $m_{\hat{B}}$) can include some values sent to $\hat{A}$ (resp., $\hat{B}$) from $\hat{B}$ (resp., $\hat{A}$), which does not affect the pre-computability of (s,r)OAKE. In particular, in practice $m_{\hat{A}}$ (resp., $m_{\hat{B}}$) can include a random nonce generated and sent by $\hat{B}$ (resp., $\hat{A}$).

In the following, we show the security of (s)OAKE-HDR (resp., rOAKE-HDR), with off-line pre-computed DH-exponents, DH-components, and the values A^{yc} or B^{xd} (resp., only off-line pre-computed DH-exponents, DH-components) that may be potentially exposed to the forger *even prior to the session involving these pre-computed values*, on which the security of OAKE and sOAKE in the CK-framework will be based. In particular, we show that our OAKE-HDR and sOAKE-HDR satisfy a stronger security definition (than the definition given in [42]). As the analysis of (s)OAKE-HDR and rOAKE-HDR assumes different secrecy exposure, we prove the security of them separately.

G.1.1 Analysis of (s)OAKE-HDR

Definition G.2 (*Strong* security of HDR signatures (with off-line pre-computation)) *We say a HDR signature scheme (of $\hat{B}$) is strongly secure, if no polynomial-time machine $\mathcal{F}$ can win the game in Figure 2 with non-negligible probability* with respect to any uncorrupted party $\hat{A}$ of public-key $A = g^a$ such that the secret-key a was not chosen by the attacker $\mathcal{F}$.

On the *strong* security of HDR. The *strong* security of our definition for (s)OAKE-HDR lies in that:

- We assume (y, Y, A^{cy}) are off-line pre-computed, and the forger can get them prior to the session run involving them.

  This particularly renders stronger capability to the attacker to perform colliding (birthday) attacks against the hash function h (that is of length $|q|/2$ for HMQV). To deal with this subtlety, the actual HMQV implementation needs some changes in practice (to be clarified later).

- In the forging game defined in Figure 2, the successful forgery requires that the *whole* vector $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the responses of $\hat{B}$ to $\mathcal{F}$'s queries. The definition for the security of HMQV-HCR in [42] only requires that the pair (Y_0, m_0) did not appear in responses from the signer. As we shall see below, the HMQV-HDR scheme may not be strongly secure in general.

OAKE-HDR vs. HMQV-HDR. In [42], the HMQV-HDR (of $\hat{B}$) is defined to be $\{X, Y, DSIG^{HMQV}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y) = H_K((XA^d)^{y+be})\}$, where $d = h(m_{\hat{A}}, X)$, $e = h(m_{\hat{B}}, Y)$. For building

1. Forger $\mathcal{F}$ is given values B, X_0, where $B, X_0 \in_{\mathrm{R}} G$.
2. $\mathcal{F}$ is given access to a signing oracle $\hat{B}$ (of public-key $B = g^b$ and secret-key b).
3. Each signature query from $\mathcal{F}$ to $\hat{B}$ consists of the following interactions:
   (a) $\mathcal{F}$ presents $\hat{B}$ with messages $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}})$. Here, $\hat{Z}$ can be any (even corrupted) party chosen by $\mathcal{F}$, and $Z = g^z \in G \setminus 1_G$ is the public-key of $\hat{Z}$. Note that $\mathcal{F}$ may not necessarily know the corresponding secret-key z of $\hat{Z}$.
   (b) $\hat{B}$ generates $y \in_{\mathrm{R}} Z_q^*$ and $Y = g^y$, and computes Z^{cy}, where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$ for OAKE-HDR or $c = 1$ for sOAKE-HDR. Then, $\hat{B}$ responds with $(y, Y = g^y, Z^{cy})$ to $\mathcal{F}$ (*which captures the powerful exposure capability to the forger*), and stores the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, Z^{cy})$ as an "incomplete session". *Here,* (y, Y, Z^{cy}) *can be offline pre-computed by* $\hat{B}$*, and leaked to* $\mathcal{F}$ *prior to the session involving* (y, Y, Z^{cy}).
   (c) $\mathcal{F}$ presents $\hat{B}$ with $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$, and a *challenge* X.
   (d) $\hat{B}$ checks that $X \in G \setminus 1_G$ (if not, it aborts) and that $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$ is in one of its incomplete sessions (if not, it ignores). $\hat{B}$ then computes $r = H_K(Z^{cy}X^{db+ey})$, where $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$ for OAKE-HDR (resp., $d = 1$ and $e = (m_{\hat{Z}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$ for sOAKE-HDR). $\hat{B}$ responds $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, Y, r)$ to $\mathcal{F}$, and marks the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, Z^{cy})$ as a "*complete session*", and stores with it the signature values $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, r)$.
4. $\mathcal{F}$ is allowed a polynomial number of adaptive queries to $\hat{B}$ in *arbitrarily* interleaved order.
5. $\mathcal{F}$ halts with output "fail" or with a *guess* in the form of a tuple $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$. $\mathcal{F}$'s guess is called a successful *forgery* if the following two conditions hold:
   (a) $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$ is a valid HDR-signature of $\hat{B}$ on the messages $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, where $\hat{A}$ is an uncorrupted player of public-key $A = g^a$, m_1 corresponds to $m_{\hat{A}}$ (that is an arbitrary message sent by the adversary $\mathcal{F}$ impersonating the signer $\hat{B}$ to the honest player $\hat{A}$), and m_0 corresponds to $m_{\hat{B}}$ (that is chosen by the honest player $\hat{A}$). Note that the value X_0 is the one received by $\mathcal{F}$ as input.
   (b) $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ *did not appear in any one of the responses of* $\hat{B}$ *to* $\mathcal{F}$*'s queries.*

   We say $\mathcal{F}$ wins the game, if it outputs a successful forgery (w.r.t. any uncorrupted player $\hat{A}$).

Figure 2: Forgery game for (strongly secure) (s)OAKE-HDR signatures (with offline pre-computation)

HMQV with HMQV-HDR, $m_{\hat{B}}$ (resp., $m_{\hat{A}}$) is set to be its *peer's* identity $\hat{A}$ (resp., $\hat{B}$). The underlying HMQV-XCR-signature is defined to be X^{y+be}, where $e = h(m_{\hat{B}}, Y)$. The following are some brief comparisons between OAKE-HDR and HMQV-HDR:

- One notable advantageous feature of (s,r)OAKE-HDR vs. HMQV-HDR is the online efficiency. Specifically, the online efficiency of (s,r)OAKE-HDR, for each player, can be only one exponentiation. In comparison, each player of HMQV-HDR performs about 1.3 online exponentiations.

- As we shall see, (s)OAKE-HDR is *strongly* secure in accordance with Definition G.2. We note that the HMQV-XCR underlying HMQV-HDR is not *strongly* secure. For example, to forge a HMQV-XCR signature $(X, Y, \sigma = X^{b+ey})$ on message m, where $e = h(m, Y)$, the forger can first query the signer with $(m, X' = X^2)$, gets back $(X', Y, \sigma' = X'^{b+ey})$, and then outputs $(X, Y, \sigma = \sigma'^{\frac{1}{2}})$ as the XCR signature on m. Note that the triple (X, Y, σ) did not appear in any one of the responses from the HMQV-XCR signer $\hat{B}$. We note that one way to remedy this vulnerability of HMQV-XCR is to commit X also to e by defining $e = h(m, X, Y)$.

- The security of (s)OAKE-HDR against uncorrupted players *other than the signer itself,* with offline pre-computed (y, Y, A^{cy}) that can be exposable to the adversary even prior to the session involving (y, Y, A^{cy}), is based on merely the GDH assumption. The security of HMQV-HDR in this case is based on both the GDH assumption and the non-standard KEA assumption [19], even if the pre-computed DH-exponent y is not exposable and only the pre-computed DH-component Y is exposable. Furthermore, as suggested in [42], for robust security of HMQV-HDR with pre-computed DH-components when the number of messages in the system is large, HMQV-HDR needs to make the following modifications : (1) Increase the output length, i.e., l, of the hash function h, e.g., from $|q|/2$ to $|q|$, which may bring negative impact on the performance of HMQV. (2) Add random nonces into the input of d and e, or, put the message to be signed also into H_K, which may increase the system complexity.

- The generation of the sOAKE-HDR signature uses minimal (i.e., only one) hashing operation (in computing the value of e).
- The HMQV-HDR signature is actually an XCR signature w.r.t. the challenge XA^d. In comparison, (s,r)OAKE-HDR in general cannot be viewed as a structure of XCR w.r.t. some challenge $f(X, A)$ for some function f.
- As we shall see, the special protocol structure of (s,r)OAKE-HDR also much simplifies, in certain scenarios, the security analysis of (s,r)OAKE in the CK-framework.

Now, we proceed to prove the security of (s,r)OAKE-HDR signatures. As we shall see, the security of (s,r)OAKE-HDR holds also w.r.t. *public-key free* versions where the public-keys A and B are removed from the input of c, d, e. But, putting the public-keys into the input of c, d, e is useful for the security of (s,r)OAKE beyond the CK-framework (e.g., security in the public computation model, session-key computational fairness, etc).

Theorem G.1 *Under the GDH assumption, (public-key free) (s)OAKE-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), are* strongly *secure in the random oracle model, with respect to any uncorrupted player other than the signer $\hat{B}$ itself* even if the forger is given the private keys of all uncorrupted players in the system other than the secret-key b of $\hat{B}$.

Proof (of Theorem G.1). Given an efficient and successful forger $\mathcal{F}$ against (s)OAKE-HDR, i.e., $\mathcal{F}$ wins the forgery game in Figure 2 with respect to some uncorrupted player $\hat{A} \neq \hat{B}$ with non-negligible probability, we build an efficient solver $\mathcal{C}$ for the GDH problem also with non-negligible probability. The algorithm $\mathcal{C}$ for OAKE-HDR is presented in Figure 3 (page 26), and the algorithm $\mathcal{C}$ for sOAKE-HDR is presented in Figure 4 (page 27).

Below, we focus on the analysis of OAKE-HDR. The analysis of sOAKE-HDR is similar and is actually simpler. For the description of $\mathcal{C}$ in Figure 3, suppose $\mathcal{F}$ makes Q_h RO queries to h, Q_H queries to H_K, Q_s signing oracle queries, where Q_h, Q_H, Q_s are polynomial in the security parameter l (i.e., the output length of h). We have the following observations:

- The signature simulation at steps S1-S3 is perfect.
- Now, suppose $\mathcal{F}$ outputs a successful forgery $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, which particularly implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0 c_0} X_0^{bd_0 + y_0 e_0}$, $X_0 = U$, $Y_0 = g^{y_0}$, $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$. We investigate the probability that $\mathcal{C}$ aborts at step F3. We have the following observations:
  - With probability at most $\frac{1}{2^l - 1} + 2^{-k} + Q_H/2^k$, $\mathcal{F}$ can succeed with undefined any one of $\{c_0, d_0, e_0\}$. Here, $\frac{1}{2^l - 1}$ is the probability that $\mathcal{F}$ guesses σ_0 with undefined c_0 or d_0 or e_0, 2^{-k} is the probability that $\mathcal{F}$ simply guesses the value r_0, and $Q_H/2^k$ is the probability upper-bound that $r_0 = H_K(\sigma_0)$ collides with some H_K-answers.
  - With defined c_0 and d_0 and e_0, there are two cases for $\mathcal{F}$ to succeed without querying $H_K(\sigma_0)$:

    Case-1. $\mathcal{F}$ simply guesses the value r_0. This probability is 2^{-k}.

    Case-2. r_0 is the value r set by $\mathcal{C}$ at one of S3.1 steps, where r is supposed to be $H_K(\sigma)$ w.r.t. a stored vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$. Recall that for the value r set at step S3.1, $\mathcal{C}$ does not know σ (as it does not know b), and thus in this case both $\mathcal{C}$ and $\mathcal{F}$ may not make the RO-query $H_K(\sigma_0) = H_K(\sigma)$. In this case, by the birthday paradox with probability at least $1 - Q_H^2/2^{-k}$, $\sigma_0 = \sigma$, i.e., $A^{c_0 y_0} X_0^{d_0 b + e_0 y_0} = Z^{cy} X^{db + ey}$,

Building the CDH solver $\mathcal{C}$ from the OAKE-HDR forger $\mathcal{F}$

Setup: The inputs to $\mathcal{C}$ are random elements $U = g^u, V = g^v$ in G, and its goal is to compute $CDH(U, V) = g^{uv}$ with oracle access to a DDH oracle $\mathcal{O}$. To this end, $\mathcal{C}$ sets $B = V$ and $X_0 = U$, and sets the public-keys and secret-keys for all other uncorrupted players in the system. $\mathcal{C}$ runs the forger $\mathcal{F}$ on input (B, X_0) against the signer $\hat{B}$ of public-key B. $\mathcal{C}$ provides $\mathcal{F}$ with a random tape, and provides the secret-keys of all uncorrupted players other than the signer $\hat{B}$ itself (the attacker $\mathcal{F}$ may register arbitrary public-keys for corrupted players, based on the public-keys and secret-keys of uncorrupted players).

Signature query simulation: Each time $\mathcal{F}$ queries $\hat{B}$ for a signature on values $(\hat{Z}, Z, m_{\hat{B}}, m_{\hat{A}})$, $\mathcal{C}$ answers the query for $\hat{B}$ as follows (note that $\mathcal{C}$ does not know b):

S1. $\mathcal{C}$ generates $y \in_{\mathrm{R}} Z_q^*$, $Y = g^y$ and Z^{cy}, where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$ (that may be pre-defined, otherwise $\mathcal{C}$ defines c with the RO h). Actually, (y, Y, Z^{cy}) can be pre-computed by $\mathcal{C}$ and leaked to $\mathcal{F}$ prior to the session. Then, $\mathcal{C}$ responds $(y, Y = g^y, Z^{cy})$ to $\mathcal{F}$, and stores the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, A^{cy})$ as an "incomplete session".

S2. $\mathcal{F}$ presents $\mathcal{C}$ with $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$, and a *challenge* X.

S3. $\hat{B}$ checks that $X \in G \setminus 1_G$ (if not, it aborts) and that $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$ is in one of its incomplete sessions (if not, it ignores the query). Then, $\mathcal{C}$ checks for every value $\sigma \in G \setminus 1_G$ *previously used by* $\mathcal{F}$ as input to H_K whether $\sigma = Z^{cy} X^{bd+ye}$, where $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$ (in case d, e undefined, $\mathcal{C}$ defines them with h): it does so using the DDH-oracle $\mathcal{O}$, specifically, by checking whether $CDH(X, B) = (\sigma / Z^{cy} X^{ye})^{d^{-1}}$. If the answer is positive, then $\mathcal{C}$ sets r to the already determined value of $H_K(\sigma)$.

S3.1. In any other cases, r is set to be a random value in $\{0,1\}^k$, where k is the output length of H_K. Note that, in this case, $\mathcal{C}$ does not know $\sigma = Z^{cy} X^{db+ey}$, as it does not know b, which also implies that $\mathcal{C}$ does not make (actually realize) the RO-query $H_K(\sigma)$ *even if the value σ has been well-defined and known to $\mathcal{F}$.*

Finally, $\mathcal{C}$ marks the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy})$ as a "*complete session*", stores $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$ and responds $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, Y, r)$ to $\mathcal{F}$.

RO queries: $\mathcal{C}$ provides random answers to queries to the random oracles h and H_K (made by $\mathcal{F}$), under the limitation that if the same RO-query is presented more than once, $\mathcal{C}$ answers it with the same response as in the first time. But, for each *new* query σ to H_K, $\mathcal{C}$ checks whether $\sigma = Z^{cy} X^{db+ey}$ for any one of the stored vectors $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$ (as before, this check is done using the DDH-oracle). If equality holds then the corresponding r is returned as the predefined $H_K(\sigma)$, otherwise a random r is returned.

Upon $\mathcal{F}$'s termination. When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs a valid HDR-signature $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, where $\hat{A} \neq \hat{B}$ is an uncorrupted player. In particular, it implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0 c_0} X_0^{bd_0 + y_0 e_0}$, $Y_0 = g^{y_0}$ (chosen by $\mathcal{F}$), $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$.

F2. $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the above responses of the simulated OAKE-HDR signatures.

F3. The values $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$ were queried from the RO h, and the value $H_K(\sigma_0)$ was queried from H_K *being posterior to the queries* c_0, d_0, e_0. Otherwise, $\mathcal{C}$ aborts.

If these three conditions hold, $\mathcal{C}$ proceeds to the "repeat experiments" below, else it aborts.

The repeat experiments. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time, under the same input (B, X_0) and using the same coins for $\mathcal{F}$. There are two cases according to the order of the queries of $h(m_0, \hat{B}, B, X_0)$ and $h(X_0, Y_0)$

C1. $h(m_0, \hat{B}, B, X_0)$ posterior to $h(X_0, Y_0)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(m_0, \hat{B}, B, X_0)$, responds back a new independent value $d_0' \in_{\mathrm{R}} \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0 c_0'} X_0^{bd_0' + y_0 e_0}$, $\mathcal{C}$ computes $CDH(U, V) = CDH(X_0, B) = [(\sigma_0 / Y_0^{ac_0}) / (\sigma_0' / Y_0^{a' c_0'})]^{(d_0 - d_0')^{-1}}$, where a and a' are the private keys of the *uncorrupted* $\hat{A}$ and $\hat{A}'$ (different from $\hat{B}$, which are assumed to be known to $\mathcal{C}$). Note that $(\hat{A}', A', m_1')$ need not necessarily to equal $(\hat{A}, A, m_1)$.

C2. $h(X_0, Y_0)$ posterior to $h(m_0, \hat{B}, B, X_0)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(X_0, Y_0)$, responds back a new independent value $e_0' \in_{\mathrm{R}} \{0,1\}^l$. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0 c_0'} X_0^{bd_0 + y_0 e_0'}$, $\mathcal{C}$ computes $X_0^{y_0} = ((\sigma_0 / Y_0^{ac_0}) / (\sigma_0' / Y_0^{a' c_0'}))^{(e_0 - e_0')}$, and then $CDH(U, V) = CDH(X_0, B) = (\sigma_0 / ((X_0^{y_0})^{e_0} \cdot Y_0^{ac_0}))^{d_0^{-1}}$.

Figure 3: Reduction from GDH to OAKE-HDR forgeries

Building the CDH solver $\mathcal{C}$ from the sOAKE-HDR forger $\mathcal{F}$

Setup: same as in Figure 3 for the forger $\mathcal{F}$ against OAKE-HDR.
Signature query and RO query simulation: similar to those made for the forger against OAKE-HDR in Figure 3, but with the following modifications:

- $c = d = 1$ and $e = h(m_{\hat{Z}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$.
- In Step S.3 and RO queries, the challenger $\mathcal{C}$ checks that $\sigma = Z^{cy}X^{bd+ye}$ by checking whether $CDH(X, B) = (\sigma/Z^y X^{ye})$ via the DDH oracle.

Upon $\mathcal{F}$'s termination. When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs a valid HDR-signature $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, where $\hat{A} \neq \hat{B}$ is an uncorrupted player. In particular, it implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0}X_0^{b+y_0e_0}$, $Y_0 = g^{y_0}$ (chosen by $\mathcal{F}$), and $e_0 = h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0Y_0)$.

F2. $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the above responses of the simulated sOAKE-HDR signatures.

F3. The value $e_0 = h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0Y_0)$ was queried from the RO h, and the value $H_K(\sigma_0)$ was queried from H_K *being posterior to the query* e_0. Otherwise, $\mathcal{C}$ aborts.

If these three conditions hold, $\mathcal{C}$ proceeds to the "repeat experiment" below, else it aborts.
The repeat experiment. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time, under the same input (B, X_0) and using the same coins for $\mathcal{F}$. $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0Y_0)$, responds back a new independent value $e_0' \in_{\mathrm{R}} \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0}X_0^{b+y_0e_0'}$, $\mathcal{C}$ computes $CDH(X_0, Y_0) = g^{x_0y_0} = [(\sigma_0/Y_0^a)/(\sigma_0'/Y_0^{a'})]^{(e_0-e_0')^{-1}}$, where a and a' are the private keys of the uncorrupted $\hat{A}$ and $\hat{A}'$ (different from $\hat{B}$, which are assumed to be known to $\mathcal{C}$). Note that $(\hat{A}', A', m_1')$ need not necessarily to equal $(\hat{A}, A, m_1)$. Finally, $\mathcal{C}$ computes $CDH(U, V) = CDH(X_0, B) = \sigma_0/((g^{x_0y_0})^{e_0} \cdot Y_0^a)$.

Figure 4: Reduction from GDH to sOAKE-HDR forgeries

where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$, $d = h(m_{\hat{B}}, \hat{B}, B, X)$, $e = h(X, Y)$, $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, Y_0)$, and $(m_0, m_1, \hat{A}, A, \hat{B}, B, X_0, Y_0) \neq (m_{\hat{A}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$. By the NMJPOK and TBSS properties of OAKE, for any value $\sigma \in G \setminus 1_G$ and any $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, the probability $\Pr[A^{c_0y_0}X_0^{d_0b+e_0y_0} = \sigma] \leq \frac{1}{2^l-1}$, where X_0 is the given random element in $G \setminus 1_G$, $\hat{A}$ and $\hat{B}$ are uncorrupted players. *This is true, even if the public-key A (resp., B) is removed from* c_0 *(resp.,* d_0*), as the public-keys A and B are generated by the uncorrupted players* $\hat{A}$ *and* $\hat{B}$ *independently at random.* Then, by straightforward calculation, we can get that $\mathcal{F}$ succeeds in Case-2 with probability at most $O(\frac{Q_h^2}{2^l-1} + \frac{Q_s+Q_H^2}{2^k})$.
Note: To rule out the possibility of Case-2, the analysis of HMQV-HCR requires the KEA assumption [19]. Furthermore, to resist to birthday attacks in Case-2 (when the number of messages in the system may be large), some modifications of HMQV are recommended in [42]: (1) increase the output length, i.e., l, of h, e.g., from $|q|/2$ to $|q|$. (2) Add random and *fresh* nonces (which cannot be offline pre-computed) to the input of h, or put the messages to be signed $m_{\hat{A}}, m_{\hat{B}}$ into the input of H_K. More details are referred to [42].

  - With probability at most $\frac{1}{2^l-1}$, the query $H_K(\sigma_0)$ is prior to any one of the queries $\{c_0, d_0, e_0\}$.

- It is easy to check that, in case the forger $\mathcal{F}$ successfully outputs another different forge satisfying the conditions F1-F3 in the repeat experiment C1 or C2, the output of $\mathcal{C}$ is the correct value of $CDH(X_0, B)$.

The similar observations can be easily checked for the algorithm $\mathcal{C}$ for sOAKE-HDR described in Figure 4. Putting all together, we have that: suppose for some uncorrupted player $\hat{A} \neq \hat{B}$, the forger $\mathcal{F}$ provides, with non-negligible probability, a successful forgery w.r.t. $\hat{A}$ in its real interactions with the signer of OAKE-HDR/sOAKE-HDR, then with also non-negligible probability (up to a negligible gap specified by the above observations) $\mathcal{F}$ succeeds under the run of $\mathcal{C}$. Then, by applying the forking lemma [60], the theorem is established.[5] □

On the role of putting players' public-keys into the inputs of c, d for OAKE-HDR and e for sOAKE-HDR. We remark that the players' public-keys in the inputs of c, d, e for (s)OAKE-HDR actually play *no* role in the above security analysis. That is, the above security analysis is actually with respect to a (*public-key free*) variant of (s)OAKE-HDR, with public-keys are removed from the inputs of c, d, e. Recall that, players' public-keys are only used for arguing the TBSS property of (s)OAKE-HDR. Specifically, for any value $\sigma \in G \setminus 1_G$ and any $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, the probability $\Pr[\sigma_0 = A^{c_0 y_0} X_0^{d_0 b + e_0 y_0} = \sigma] \leq \frac{1}{2^l - 1}$, where $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ $e_0 = h(X_0, Y_0)$ for OAKE-HDR (resp., $c_0 = d_0 = 1$ and $e_0 = h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$ for sOAKE-HDR), and the probability is taken over only the choice of the random function h. But, as we assume $\hat{A}$ and $\hat{B}$ are both uncorrupted players, their public-keys are generated independently at random. Also, the value X_0 is the given random DH-component (not generated by the attacker). To affect the distribution of σ_0, the only freedom of the attacker is to maliciously choose (Y_0, m_0, m_1), which however does not change the distribution of σ_0. In particular, for any value $\sigma \in G \setminus 1_G$ and for any (Y_0, m_0, m_1) chosen maliciously by the attacker w.r.t. the fixed $(\hat{A}, A, \hat{B}, B, X_0)$, it still holds that $\Pr[\sigma_0 = \sigma] \leq \frac{1}{2^l - 1}$. But, putting the public-keys into the input of c, d, e is useful for the security of (s,r)OAKE beyond the CK-framework (e.g., security in the public computation model, session-key computational fairness, etc).

Security of (s)OAKE-HDR against the signer itself. The above security analysis considers the security of (s)OAKE-HDR against any other uncorrupted players other than the signer itself, i.e., the (in)feasibility of outputting a successful forgery $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0, r_0)$ where $\hat{A}$ is an uncorrupted player and $\hat{A} \neq \hat{B}$. But, the forger $\mathcal{F}$ may also be against the signer $\hat{B}$ itself. That is, $\mathcal{F}$ may output a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, Y_0, r_0)$ (i.e., $\hat{A} = \hat{B}$). Here, we further investigate the feasibility of successful forgeries of this form. We distinguish two cases: (1) $Y_0 = X_0$, i.e., the successful forgery is of the form $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$. For this case, similar to that of HMQV-HDR, we show (s)OAKE-HDR is secure under the traditional CDH assumption (not the stronger GDH assumption) in the RO model; (2) $Y_0 \neq X_0$. For this case, we show (s)OAKE-HDR is secure under the GDH assumption, and additionally *the KEA assumption*, in the RO model. We remark that the KEA assumption is only used to rule out the feasibility of successful forgeries in the special case of $Y_0 \neq X_0$ and $\hat{A} = \hat{B}$.

Corollary G.1 Under the computational Diffie-Hellman (CDH) assumption, *(public-key free) (s)OAKE-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), are* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself and $Y_0 = X_0$.*

Proof. This case implies that the forger $\mathcal{F}$ can output, with non-negligible probability, a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$, where $r_0 = H_K(\sigma_0)$, $\sigma_0 = B^{c_0 x_0} X_0^{d_0 b + e_0 x_0} = (X_0^{c_0} X_0^{d_0})^b X_0^{e_0 x_0}$, $c_0 = h(m_1, \hat{B}, B, X_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, X_0)$ for OAKE-HDR (for sOAKE-HDR, $c_0 = d_0 = 1$ and $e_0 = h(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0)$). Note that from σ_0 and $\hat{B}$'s secret-key b, we can compute $X_0^{x_0}$. But, the hardness of computing X^x from random X is equivalent to that of the CDH problem [48, 53].

With the above observations, we modify the algorithm $\mathcal{C}$ depicted in Figure 3 and Figure 4 as follows:

- $\mathcal{C}$ knows (sets) also the private key b for $\hat{B}$. By knowing the private key b, $\mathcal{C}$ dispenses with the DDH-oracle in order to make the answers to signature queries and RO-queries to be consistent.

[5] We note that we can also use the more general and abstract forking lemmas [1, 5], but with a slightly different security argument procedure.

- After $\mathcal{F}$ outputs a successful forgery of the form $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$, satisfying the conditions F1-F3, $\mathcal{C}$ simply computes out $X_0^{x_0}$ from σ_0 and the private-key b. Note that $\mathcal{C}$ does not need to perform the rewinding experiments at all in this case. □

Now we consider the case of $Y_0 \neq X_0$. As mentioned, it is the only place we need to additionally use the KEA assumption.

Definition G.3 *[Knowledge-of-Exponent Assumption (KEA)] Let G be a cyclic group of prime order q generated by an element g, and consider algorithms that on input a triple $(g, C = g^c, z)$ output a pair $(Y, Z) \in G^2$, where c is taken uniformly at random from Z_q^* and $z \in \{0,1\}^*$* is an arbitrary string that is generated independently of C. *Such an algorithm $\mathcal{A}$ is said to be a KEA algorithm if with non-negligible probability (over the choice of g, c and $\mathcal{A}$'s random coins) $\mathcal{A}(g, g^c, z)$ outputs $(Y, Z) \in G^2$ such that $Z = Y^c$. Here, $C = g^c$ is the random challenge to the KEA algorithm $\mathcal{A}$, and z captures the auxiliary input of $\mathcal{A}$ that is independent of the challenge C.*

We say that the KEA assumption holds over G, if for every probabilistic polynomial-time (PPT) KEA algorithm $\mathcal{A}$ for G there exists another efficient algorithm $\mathcal{K}$, referred to as the KEA-extractor, for which the following property holds except for a negligible probability: let (g, g^c, z) be an input to $\mathcal{A}$ and ρ a vector of random coins for $\mathcal{A}$ on which $\mathcal{A}$ outputs $(Y, Z = Y^c)$, then, on the same inputs and random coins, $\mathcal{K}(g, C, z, \rho)$ outputs the triple $(Y, Z = Y^c, y)$ where $Y = g^y$.

Corollary G.2 *Under the GDH assumption, and additionally the KEA assumption, (public-key free) (s)OAKE-HDR and sOAKE-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), is* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself and $Y_0 \neq X_0$.*

Proof. The proof of Corollary G.2 follows the same outline of that of Theorem G.1. We highlight the main differences, and how the KEA assumption comes into force in the security analysis. The analysis is mainly w.r.t. OAKE-HDR (the similar, and actually simpler, holds also for sOAKE-HDR).

The main difference between the proof of Corollary G.2 and that of Theorem G.1 is that, here, the forger outputs with non-negligible probability a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, Y_0, r_0)$, where $r_0 = H_K(\sigma_0)$, $\sigma_0 = B^{c_0 y_0} X_0^{d_0 b + e_0 y_0}$, $c_0 = h(m_1, \hat{B}, B, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, Y_0)$ (for sOAKE-HDR, $c_0 = d_0 = 1$ and $e_0 = h(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0)$). The key point is that, by performing the rewinding experiments, we cannot directly output the $CDH(B, X_0)$, as we do not know the private key b of $\hat{B}$ (recall that we are going to compute $CDH(B, X_0)$ by running the forger $\mathcal{F}$). Note that in the security analysis of Theorem G.1, we heavily relied on the fact that we know the private key of any uncorrupted player other than the signer itself.

We modify the algorithm $\mathcal{C}$ depicted in Figure 3 as follows: the actions of $\mathcal{C}$ remain unchanged until the rewinding experiments; $\mathcal{C}$ performs the rewinding experiments according to the order of the RO-queries c_0, d_0, e_0.

d_0 **posterior to** c_0, e_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $d_0 = h(m_0, \hat{B}, B, X_0)$, and redefines $h(m_0, \hat{B}, B, X_0)$ to be a new independent d'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c_0 y_0} X_0^{d'_0 b + e_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets that $CDH(B, X_0) = (\sigma/\sigma'_0)^{(d_0 - d'_0)^{-1}}$. *Note that, in this case, $\mathcal{C}$ does not rely on the KEA assumption for breaking the CDH assumption* (but still with the DDH-oracle).

c_0 **posterior to** d_0, e_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $c_0 = h(m_1, \hat{B}, B, Y_0)$, and redefines $h(m_1, \hat{B}, B, Y_0)$ to be a new independent c'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c'_0 y_0} X_0^{d_0 b + e_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets $CDH(B, Y_0) = B^{y_0} = (\sigma/\sigma'_0)^{(c_0 - c'_0)^{-1}}$. That is, given B, $\mathcal{C}$ can output (Y_0, B^{y_0}). By the KEA assumption, it implies that $\mathcal{F}$ knows y_0 (which can be derived from the internal state of $\mathcal{F}$). More formally, there exists an algorithm that, given B and X_0 and the random coins of $\mathcal{C}$ and $\mathcal{F}$ can successfully output y_0. Now, with the knowledge of y_0, $CDH(B, X_0)$ can be derived from σ_0 (or σ'_0).

e_0 **posterior to** c_0, d_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $e_0 = h(X_0, Y_0)$, and redefines $h(X_0, Y_0)$ to be a new independent e'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c_0 y_0} X_0^{d_0 b + e'_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets $CDH(X_0, Y_0) = X_0^{y_0} = (\sigma/\sigma'_0)^{(e_0 - e'_0)^{-1}}$. Then, by the KEA assumption, the knowledge of y_0 can be derived, with which $CDH(X_0, B)$ can then be computed from either σ_0 or σ'_0.

The analysis of sOAKE-HDR in this case is simpler. By redefining $h(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0)$ to be a random value $e'_0 \neq e_0$, $\mathcal{C}$ will gets $\sigma'_0 = B^{c_0 y_0} X_0^{d_0 b + e'_0 y_0}$, where $c_0 = d_0 = 1$. From (σ_0, σ'_0), $\mathcal{C}$ can output $CDH(X_0, Y_0) = (\sigma/\sigma'_0)^{(e_0 - e'_0)^{-1}}$. Then, by the KEA assumption, the knowledge of y_0 can be derived, with which $CDH(X_0, B)$ can then be computed from either σ_0 or σ'_0. □

G.1.2 Analysis Extension and Adaptation to rOAKE-HDR

In this section, we show how the analysis of (s)OAKE-HDR signatures presented in Section G.1.1 can be extended and adapted to that of rOAKE-HDR without relying on the non-standard KEA assumption. In the following security analysis, we assume only the values (y, Y) (resp., (x, X)) can be exposed for the security of $\hat{B}$ (resp., $\hat{A}$). Accordingly, in the forgery game of HDR signature described in Figure 2, only the values (y, Y) are exposed to the attacker against the signer $\hat{B}$ (and the value Z^{cy} is removed throughout the forgery game in Figure 2). Note that, for rOAKE, the values A^{b+yc} and B^{a+xd} can still be offline pre-computed. But, for the security of rOAKE in the CK-framework, these pre-computed values should be well-protected and should not be exposed to attacker. This is contrary to (s)OAKE, where all the pre-computed values (y, Y, A^{cy}) can be exposed.

Corollary G.3 *Under the GDH assumption, (public-key free) rOAKE-HDR signatures of $\hat{B}$, with exposed (y, Y), are strongly secure in the random oracle model, with respect to any uncorrupted player other than the signer $\hat{B}$ itself.*

Proof. The proof follows the same outline of that of Theorem G.1. Here, we mainly highlight the key differences between them.

In Step S.1 the challenger $\mathcal{C}$ couldn't provide the value Z^{bf+cy}, as it does not know $\hat{B}$'s secret-key b. This is also the reason that we assume only the values (y, Y) can be exposed.

In Step S.3 and RO queries, the challenger $\mathcal{C}$ checks that $\sigma = Z^{b+cy} X^{bd+ye}$ by checking whether $CDH(X^d Z, B) = (\sigma / Z^{cy} X^{ye})$ via the DDH oracle.

In steps F1-F3, denote $\sigma_0 = A^{b+y_0 c_0} X_0^{bd_0 + y_0 e_0}$, where $Y_0 = g^{y_0}$ is chosen by $\mathcal{F}$, $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$.

For Case C1 in the repeat experiment (i.e, d_0 posterior to e_0), suppose $\sigma'_0 = A'^{b + y_0 c'_0} X_0^{bd'_0 + y_0 e_0}$, where $c'_0 = h(m'_1, \hat{A}', A', Y_0)$, $d'_0 = h(m_0, \hat{B}, B, X_0) \neq d_0$, $e_0 = h(X_0, Y_0)$, and $\hat{A}'$ is another uncorrupted player that is different from $\hat{B}$ and A' is its public-key. Note that, as the secret-keys of all uncorrupted players except $\hat{B}$ are set by $\mathcal{C}$ and both $\hat{A}$ and $\hat{A}'$ are different from $\hat{B}$, the secret-keys of $\hat{A}$ and $\hat{A}'$, denoted a and a' respectively, are known to $\mathcal{C}$. Then, from $(\sigma_0, \sigma'_0, a, a')$, $\mathcal{C}$ can compute $CDH(U, V) = CDH(X_0, B) = [(\sigma_0/(Y_0^{c_0} B)^a)/(\sigma'_0/Y_0^{c'_0} B)^{a'_0})]^{(d_0 - d'_0)^{-1}}$.

For Case C2 in the repeat experiment (i.e, e_0 posterior to d_0), suppose $\sigma'_0 = A'^{b + y_0 c'_0} X_0^{bd_0 + y_0 e'_0}$, where $c'_0 = h(m'_1, \hat{A}', A', Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e'_0 = h(X_0, Y_0) \neq e_0$. From σ_0 and σ'_0, $\mathcal{C}$ can compute $CDH(X_0, Y_0) = [(\sigma_0/(Y_0^{c_0} B)^a)/(\sigma'_0/Y_0^{c'_0} B)^{a'_0})]^{(e_0 - e'_0)^{-1}}$, and then $CDH(U, V) = CDH(X_0, B) = [\sigma_0/((Y_0^{c_0} B)^a CDH(X_0, Y_0)^{e_0})]^{d_0^{-1}}$. □

The security analysis for the case of $\hat{A} = \hat{B}$ and $X_0 = Y_0$ is just the same as that of Corollary G.1. In this case, the challenger $\mathcal{C}$ sets secret-keys for all uncorrupted players (including, in particular, the signer $\hat{B}$), and reduce the HDR forgery to the ability of computing $CDH(X_0, X_0)$ (that is equal to the ability of breaking the traditional CDH assumption). Here, as $\mathcal{C}$ knows the secret-key b of $\hat{B}$, $\mathcal{C}$ can provide the pre-computed value Z^{b+cy} to the attacker. We have the following corollary:

Corollary G.4 *Under the CDH assumption, (public-key free) rOAKE-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{b+cy}), are* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself and $Y_0 = X_0$.*

Now, we consider the case of $\hat{A} = \hat{B}$ but $X_0 \neq Y_0$. For this case, the proof of Corollary G.2 can still be extended to rOAKE-HDR under both the GDH and KEA assumptions, *assuming only (y, Y) can be exposed.* The key difference, in comparison with the proof of Corollary G.2, is that: for rOAKE-HDR signature, the output of the challenger $\mathcal{C}$ during the rewinding experiments is $CDH(B, X_0)$ for the case of d_0 posterior to c_0 and e_0, and is $CDH(X_0^{d_0}B, B) = X_0^{bd_0}B^b$ in the rest two cases. But, it is easy to check that, given random elements $B = g^b, X = g^x \leftarrow G \setminus 1_G$, the hardness of computing $CDH(X^dB, B) = (X^dB)^b$ is equivalent to that of computing $CDH(B, X)$.

Proposition G.1 *Given random elements $B = g^b, X = g^x \leftarrow G \setminus 1_G$, where $b, x \leftarrow Z_q^*$, the hardness of computing $CDH(B, X)$ is equivalent to that of computing $CDH(X^dB, B) = (X^dB)^b$, where $d = h(\hat{B}, B, X)$.*

Proof (of Proposition G.1). First recall that the hardness of computing B^b from random $B = g^b$ is equivalent to that of the CDH problem [48, 53]. Thus, the ability of computing $CDH(B, X)$ (given (B, X)) is equivalent to the ability of computing B^b (given B only), which then implies the ability of computing $CDH(X^dB, B) = X^{bd}B^b$.

Suppose there exists an efficient algorithm $\tilde{A}$ that can compute $CDH(X^d, B) = X^{db}B^b$ (from B and X) with non-negligible probability, then there exists another efficient algorithm $\tilde{B}$ that can break the CDH assumption with also non-negligible probability. The input of $\tilde{B}$ is a random element $B \in G \setminus 1_G$, and its goal is to break the CDH assumption by computing $CDH(B, B) = B^b$. Towards this goal, $\tilde{B}$ generates $X = g^x$ where x is taken uniformly at random from Z_q^*, and then runs $\tilde{A}$ on input (B, X). After getting $CDH(X^dB) = X^{db}B^b = B^{xd}B^b$ from the output of $\tilde{A}$, $\tilde{B}$ computes $B^b = CDH(X^dB, B)/B^{xd}$.
□

But, the key observation for rOAKE-HDR in this case is: *as the value g^{ab} is involved and $c = h(m_1, \hat{A}, A, Y)$, we can have an alternative way to reduce the rOAKE-HDR signature forgery to the ability of computing $CDH(B, A) = CDH(B, B)$ (rather than $CDH(X_0, B)$ as in the analysis of (s)OAKE-HDR).* Recall that $B = A$ in this case, and the hardness of computing $CDH(B, B)$ is equivalent to the standard CDH assumption. Moreover, in comparison with the proof of Corollary G.2 for (s)OAKE-HDR in this case, the analysis for rOAKE-HDR in this case is much simplified without considering the order among c_0, d_0, e_0 in the repeat experiment. Specifically, we have the following corollary:

Corollary G.5 *Under the GDH assumption, (public-key free) rOAKE-HDR signatures of $\hat{B}$, with offline pre-computed and exposed (y, Y), are* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself and $Y_0 \neq X_0$.*

Proof. The proof follows the proof procedures of Theorem G.1 and Corollary G.3. Here, we mainly highlight the key differences between them.

In the Setup, $\mathcal{C}$ only takes the value B as its input, while the value $X_0 = g^{x_0}$ is generated by $\mathcal{C}$ itself (and thus x_0 is known to $\mathcal{C}$). Then, $\mathcal{C}$ runs the forger $\mathcal{F}(B, X_0)$.

Until the repeat experiment, $\mathcal{C}$ acts just as it does for the case of $\hat{A} \neq \hat{B}$ in the proofs of Corollary G.3 and Theorem G.1. Denote by $\sigma_0 = A^{b+y_0c_0}X_0^{bd_0+y_0e_0} = B^{b+y_0c_0}X_0^{bd_0+y_0e_0}$ the value defined in Steps F1-F3, where $Y_0 = g^{y_0}$ is chosen by $\mathcal{F}$, $c_0 = h(m_1, \hat{A}, A, Y_0) = h(m_1, \hat{B}, B, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$ (recall that $\hat{A} = \hat{B}$ for this case).

In the repeat experiment, $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of just querying the RO h with $(m_1, \hat{A}, A, Y_0) = (m_1, \hat{B}, B, Y_0)$, redefines $h(m_1, \hat{B}, B, Y_0)$ to be a random value $c'_0 \neq c_0$, and runs $\mathcal{F}$ further from this rewinding point. Note that $\mathcal{C}$ does not need to consider the order of RO queries among c_0, d_0, e_0 for the repeat experiment.

Suppose $\mathcal{C}$ outputs another successful forgery, and denote by $\sigma'_0 = B^{b+y_0c'_0}X_0^{bd'_0+y_0e'_0}$ the value defined in the repeat experiment, where $c'_0 = h(m_1, \hat{B}, B, Y_0) \neq c_0$, $d'_0 = h(m'_0, \hat{B}, B, X_0)$, $e'_0 =$

$h(X_0, Y_0)$(whether (d'_0, e'_0) is identical to (d_0, e_0) or not). Note that $\mathcal{C}$ knows the value x_0 such that $X_0 = g^{x_0}$. From $(\sigma_0, \sigma'_0, x_0)$, $\mathcal{C}$ computes $CDH(B, Y_0) = [(\sigma_0/(B^{d_0}Y_0^{e_0})^{x_0})/(\sigma'_0/(B^{d'_0}Y_0^{e'_0})^{x_0})]^{(c_0-c'_0)^{-1}}$, from which $\mathcal{C}$ can then compute $CDH(B, B) = \sigma_0/[(B^{d_0}Y_0^{e_0})^{x_0}CDH(B, Y_0)^{c_0}]$. □.

G.2 SK-Security Analysis of (s,r)OAKE with Offline Pre-Computation

In the following, we first present the SK-security analysis of (s)OAKE in the CK-framework (with pre-specified peers), with offline pre-computed and exposed DH-exponents, DH-components, and DH-secrets derived from one's DH-component and its peer's public-key (say, A^{cy} and B^{dx}) which may be exposed to the adversary prior to the session involving these pre-computed values. Then, we show how the analysis can be extended and adapted to that of rOAKE.

Analysis of (s)OAKE. Using the terminology of HDR signatures, a session of OAKE (resp., sOAKE) between two parties $\hat{A}$ and $\hat{B}$ consists of a basic Diffie-Hellman exchange of DH-components $X = g^x$ and $Y = g^y$; And the session-key K is then computed as the corresponding HDR-signatures, specifically, $K = HSIG^{OAKE}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y)$ for OAKE and (resp., $K = HSIG^{sOAKE}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y)$ for sOAKE), where $m_{\hat{A}}$ *and* $m_{\hat{B}}$ *are set to be the empty string for both OAKE and sOAKE.*

During a session of (s)OAKE within the CK-framework, with offline pre-computation, a party can be activated with three types of activations (for presentation simplicity, we assume $\hat{A}$ denotes the identity of the party being activated and $\hat{B}$ the identity of the intended peer to the session):

Initiate$(\hat{A}, \hat{B})$ (i.e., $\hat{A}$ is activated as the initiator): $\hat{A}$ generates a value $X = g^x$, $x \in_{\mathrm{R}} Z_q^*$, creates a local session of the protocol which it identifies as the incomplete (open) session $(\hat{A}, \hat{B}, X)$, and outputs the DH-component X as its outgoing message.

Here (X, x, B^{dx}), where $d = h(\hat{B}, B, X)$ for OAKE or $d = 1$ for sOAKE can be offline pre-computed by $\hat{A}$, which may be exposed to the adversary prior to the session involving them.

Respond$(\hat{A}, \hat{B}, Y)$ (i.e., $\hat{A}$ is activated as the responder): $\hat{A}$ checks $Y \in G \setminus 1_G$, if so it generates a value $X = g^x$, $x \in_{\mathrm{R}} Z_q^*$, outputs X, computes the session-key and then completes the session $(\hat{A}, \hat{B}, X, Y)$.

Again, (X, x, B^{dx}) can be offline pre-computed by $\hat{A}$, which may be exposed to the adversary prior to the session involving them.

Complete$(\hat{A}, \hat{B}, X, Y)$ (i.e., the initiator $\hat{A}$ receives Y from the responder peer $\hat{B}$): $\hat{A}$ checks that $Y \in G \setminus 1_G$ and that it has an open session with identifier $(\hat{A}, \hat{B}, X)$. If any of these conditions fails $\hat{A}$ ignores the activation, otherwise it computes the session-key and completes the session $(\hat{A}, \hat{B}, X, Y)$.

With the above notation, it is ensured that if $(\hat{A}, \hat{B}, X, Y)$ is a complete session at $\hat{A}$, then its matching session (if it exists) is unique, which is $(\hat{B}, \hat{A}, Y, X)$ owned by the player $\hat{B}$. In the following analysis, we specify that the values, exposable to the adversary via session-state query (against an incomplete session), include the DH-component and DH-exponent and the DH-secret of one's DH-component and its peer's public-key, i.e., (Y, y, A^{cy}) and (X, x, B^{dx}).

Theorem G.2 *Under the GDH assumption in the RO model, the OAKE and sOAKE protocols (actually, the variants with public-keys removed from the inputs of* c, d, e*), with offline pre-computed DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say* A^{cy} *and* B^{dx}*), are SK-secure in the CK-framework w.r.t. any test-session between a pair of different players.*

Proof. According to the SK-security definition in the CK-framework, we need to prove OAKE and sOAKE satisfy the following two requirements:

Requirement-1. If two parties $\hat{A}, \hat{B}$ complete matching sessions, then their session-keys are the same.

Requirement-2. Under the GDH assumption, there is no feasible adversary that succeeds in distinguishing the session-key of an unexposed session with non-negligible probability.

The Requirement-1 can be trivially checked for both OAKE and sOAKE. In the following, we focus on establishing the Requirement-2.

Denote by $(\hat{A}, \hat{B}, X_0 = g^{x_0}, Y_0 = g^{y_0})$ the unexposed test-session between a pair of *uncorrupted* players $\hat{A}$ and $\hat{B}$ where $\hat{A} \neq \hat{B}$, and by $H_K(v_0)$ the session-key of the test-session that is referred to as the *test* HDR-signature, where $v_0 = A^{c_0 y_0} X^{d_0 b + e_0 y_0} = B^{d_0 x_0} Y^{c_0 a + e_0 x_0}$. As H_K is a random oracle, there are only two strategies for the adversary $\mathscr{A}$ to distinguish $H_K(v_0)$ from a random value:

Key-replication attack. $\mathscr{A}$ succeeds in forcing the establishment of a session (other than the test-session or its matching session) that has the same session-key output as the test-session. In this case, $\mathscr{A}$ can learn the test-session key by simply querying the session to get the same key (without having to learn the value of the test HDR-signature).

Forging attack. At some point in its run, $\mathscr{A}$ queries the RO H_K with the value v_0. This implies that $\mathscr{A}$ succeeds in computing or learning the test HDR-signature (i.e., the session-key of the test-session) via its attacks. For presentation simplicity, we assume $\mathscr{A}$ directly outputs the session-key of the test-session, referred to as the test HDR-signature, via a successful forging attack.

The possibility of key-replication attack is trivially ruled out *unconditionally* in the RO model by the TBSS property of (s)OAKE. Specifically, for any session-tag $(\hat{A}, A, \hat{B}, B, X, Y)$ and for any value $\sigma \in G \setminus 1_G$, the probability $\Pr[K_{\hat{A}} = K_{\hat{B}} = \sigma] \leq \frac{1}{2^{l}-1}$ holds for both OAKE and sOAKE, where the probability is taken over only the choice of the random function h. Then, by the birthday paradox, any efficient attacker can succeed in the key-replication attack only with negligible probability (say, with probability at most $\frac{s^2}{2^{l}-1}$ by the birthday paradox, where s is the number of sessions in the system). Actually, as the test-session and its matching session are defined without taking public-keys into account in the CK-framework, the possibility of key-replication attack is trivially ruled out *unconditionally* in the RO model also for the public-key free variant of (s)OAKE. Specifically, for any test-session $(\hat{A}, \hat{B}, X, Y)$ and any session $(\hat{A}', \hat{B}', X', Y')$ that is unmatched to the test-session (which implies that at least one of the following inequalities holds: $\hat{A} \neq \hat{A}'$, $\hat{B} \neq \hat{B}'$, $X \neq X'$ and $Y \neq Y'$), it holds that $\Pr[K_{\hat{A}} = K_{\hat{A}'}] = \frac{1}{2^{l}-1}$. As the attacker is polynomial-time, it cannot make two unmatched sessions to output the same session-key with non-negligible probability.[6]

Then, in the following analysis, we only focus on ruling out the forging attack. Recall that $\hat{A} \neq \hat{B}$ for the test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ held by $\hat{A}$. In the rest, we make analysis mainly with respect to the OAKE protocol, the similar and actually simpler holds also for sOAKE.

Now, suppose there is an efficient KE-attacker $\mathscr{A}$ who succeeds, by forging attacks, against the test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ with $\hat{A} \neq \hat{B}$ (particularly, $A \neq B$), we present an efficient forger $\mathcal{F}$ against the underlying OAKE-HDR signature, which contradicts the security of the underlying OAKE-HDR signature scheme (that is based on the GDH assumption), and thus establishing the theorem. $\mathcal{F}$ works as follows, by running $\mathscr{A}$ as a subroutine.

1. The inputs of $\mathcal{F}$ are (B, X_0), and $\mathcal{F}$ has oracle access to the OAKE-HDR signer $\hat{B}$ of public-key B.

2. We assume $\mathcal{F}$ successfully guessed the *unexposed* test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ held at $\hat{A}$, where $\hat{A} \neq \hat{B}$ and $\mathcal{F}$ sets the DH-component from $\hat{A}$ for the test-session just to be X_0.

3. $\mathcal{F}$ sets the inputs to all parties other than $\hat{B}$, and thus can perfectly emulate these parties. In particular, $\mathcal{F}$ can deal with state-reveal queries, session-key queries by $\mathscr{A}$ on any session other than the test-session and its matching session, and party corruption queries on any party other than $\hat{A}$ and $\hat{B}$.

[6] In comparison, the analysis of HMQV to rule out key-replication attack in [42] is quite complicated, and is still reduced to the underlying hardness assumptions (to be precise, to the unforgeability of HMQV-HDR).

4. When $\mathscr{A}$ activates a session at $\hat{B}$, either as a responder or initiator, with peer identity $\hat{P}$ of public-key P and incoming message X, then $\mathcal{F}$ feeds $\hat{B}$ the value $(\hat{P}, P, X)$. In response, $\mathcal{F}$ gets values (y, Y, P^{cy}) from $\hat{B}$, and then $\mathcal{F}$ hands $\mathscr{A}$ the value Y as the outgoing message from $\hat{B}$. Actually, the values (y, Y, P^{cy}) can be offline pre-computed by $\hat{B}$, and leaked to $\mathcal{F}$ (and $\mathscr{A}$) prior to the session involving them.

5. When $\mathscr{A}$ issues a state-reveal query against an incomplete session $(\hat{B}, \hat{P}, Y)$ (*not matching to the test-session*) held at $\hat{B}$, then $\mathcal{F}$ returns the values (Y, y, P^{cy}) to $\mathscr{A}$.

6. When $\mathscr{A}$ issues a session-key query to a session $(\hat{B}, \hat{P}, Y, X)$ (*not matching to the test-session*) held at $\hat{B}$, then $\mathcal{F}$ queries the session-signature from its signing oracle $\hat{B}$ by presenting the signing oracle with $(\hat{P}, P, X, Y)$, and returns the HDR-signature from $\hat{B}$ to $\mathscr{A}$.

7. When $\mathscr{A}$ halts with a valid test-signature, denoted σ_0, $\mathcal{F}$ stops and outputs σ_0.

Suppose there are n parties in total in the system, and each party is activated at most m times (where n and m are polynomials in the security parameter), in actual analysis $\mathcal{F}$ guesses the test-session by choosing uniformly at random a triple $(\hat{P}_i, \hat{P}_j, t)$ (hoping that $\hat{P}_i = \hat{A}$ and $\hat{P}_j = \hat{B}$ and the test-session is the t-th session activated at $\hat{A}$ with peer $\hat{B}$), where $1 \leq i \neq j \leq n$ and $1 \leq t \leq m$. Thus, with probability $(n^2 m)^{-1}$, $\mathcal{F}$ successfully guesses the test-session. It is easy to check that, conditioned on $\mathcal{F}$ successfully checks the test-session, the view of $\mathscr{A}$ under the run of $\mathcal{F}$ is identical to that in the real run of $\mathscr{A}$ in accordance with the forgery game defined in Figure 2 . Suppose $\mathscr{A}$ successfully outputs, with non-negligible probability ε, the valid test-signature via forging attack in its real run, with still non-negligible probability $(n^2 m)^{-1}\varepsilon$ $\mathscr{A}$ (and thus $\mathcal{F}$) outputs the valid test-signature under the run of $\mathcal{F}$.

We need then to check whether the valid test HDR-signature outputted by $\mathcal{F}$ is a *successful* OAKE-HDR forgery. As the test-signature output by $\mathscr{A}$ is valid, according to Definition G.2, we only need to show the vector $\{\hat{A}, A, X_0, Y_0\}$ did not appear in any one of the responses from the signing oracle $\hat{B}$. We distinguish three cases, according to the appearance of Y_0:

Case-1. Y_0 was never output in any one of the signatures issued by $\hat{B}$. In this case, the test HDR-signature output by $\mathscr{A}$ (and thus $\mathcal{F}$) is clearly a successful forgery against OAKE-HDR.

Case-2. Y_0 was output in one of the signatures issued by $\hat{B}$ in a session *non-matching* to the test-session. Denote by $(\hat{B}, \hat{P}, Y_0, X)$ this non-matching session, we have that $\hat{P} \neq \hat{A}$ or $X \neq X_0$. That is, $(\hat{P}, P, X) \neq (\hat{A}, A, X_0)$. As $\hat{B}$ uses random and independent DH-components in each session, with overwhelming probability the value Y_0 is only used in this non-matching session $(\hat{B}, \hat{P}, Y_0, X)$, and thus does not appear (except for a negligible probability of accidental repetition) in any other signatures issued by $\hat{B}$ in other sessions different from $(\hat{B}, \hat{P}, Y_0, X)$. Thus, $\{\hat{A}, A, X_0, Y_0\}$ did not appear in any of the HDR-signatures issued by $\hat{B}$, and thus the test HDR-signature output by $\mathcal{F}$ is a successful forgery against OAKE-HDR.

Case-3. Y_0 was generated by $\hat{B}$ in the *matching* session $(\hat{B}, \hat{A}, Y_0, X_0)$. However, this matching session was never queried by $\mathscr{A}$ via session-key query or session-state query (recall we assume the test-session and its matching session are unexposed in the CK-framework), which in turn implies that $\mathcal{F}$ never queries $\hat{B}$ for the HDR-signature of this matching session. Also, the random value Y_0 was used by $\hat{B}$ only for this matching session (except for a negligible probability of accidental repetition). This implies that, in Case-3, the values $\{\hat{A}, A, X_0, Y_0\}$ also did not appear in any one of the responses from the signing oracle $\hat{B}$, and thus the test HDR-signature output by $\mathcal{F}$ is a successful forgery against OAKE-HDR. □

Notes on the security analysis of (s)OAKE in the CK-framework. For the above security analysis of (s)OAKE in the CK-framework, we have the following observations and notes:

- The analysis is actually w.r.t. the *public-key free* variants of (s)OAKE, with players' public-keys removed from the inputs of the functions of c, d, e. The reason is that: (1) public-keys are unnecessary for ruling out the key-replication attack; (2) for ruling out the forging attack, the security of the underlying (s)OAKE-HDR signatures also does not rely on them.

- The analysis shows that OAKE and sOAKE remain their security in the CK-framework, even if the attacker $\mathscr{A}$ exposes the private values (y, A^{cy}) of the matching session (but not the session-key itself). This provides extra security guarantee of (s)OAKE that is beyond the CK-framework. The reason is that, even if these pre-computed private values are used by $\hat{B}$ in the matching session $(\hat{B}, \hat{A}, Y_0, X_0)$ and exposed to $\mathscr{A}$, the forger $\mathcal{F}$ never queries the HDR-signature corresponding to this matching session as the underlying attacker $\mathscr{A}$ is not allowed to make the session-key query against the matching session (note that $\mathcal{F}$ queries the HDR signer for a session-signature only when $\mathscr{A}$ makes the session-key query against this session), and thus $(\hat{A}, A, X_0, Y_0)$ still did not appear in any one of the signatures issued by $\hat{B}$.

Using Corollary G.1 and Corollary G.2, we have the following corollaries about the security of (s)OAKE in the CK-framework w.r.t. any test-session between the identical players $\hat{A} = \hat{B}$. The proofs are straightforward adaptations of the proof of Theorem G.2, and details are omitted here.

Corollary G.6 *Under the CDH assumption in the RO model, the OAKE and sOAKE protocols (actually, the variants with public-keys removed from the inputs of c, d, e), with offline pre-computed and exposable DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say A^{cy} and B^{dx}), are SK-secure in the CK-framework w.r.t. any test-session of identical peer and identical DH-component (i.e., $\hat{A} = \hat{B}$ and $X = Y$).*

Corollary G.7 *Under the GDH assumption and additionally the KEA assumption in the RO model, the OAKE and sOAKE protocols (actually, the variants with public-keys removed from the inputs of c, d, e), with offline pre-computed and exposable DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say A^{cy} and B^{dx}), are SK-secure in the CK-framework w.r.t. any test-session of identical peer but different DH-components (i.e., $\hat{A} = \hat{B}$ but $X \neq Y$).*

Extension and adaptation to SK-security analysis of rOAKE. The SK-security analysis follows the same outline of that of OAKE. Here, we mainly highlight the major differences between them, besides the difference that the value $K_{\hat{A}} = K_{\hat{B}}$ is additionally multiplied by g^{ab} in rOAKE. We consider the three cases regarding the test-session $(\hat{A}, \hat{B}, X_0, Y_0)$. For the case of $\hat{A} \neq \hat{B}$, the SK-security analysis of rOAKE is essentially identical to that of (s)OAKE, with the following modification: only the values (X, x) and (Y, y) can be exposed for an incomplete session. For the case of $\hat{A} = \hat{B}$ but $X_0 \neq Y_0$, there is an additional difference between the SK-security analysis of rOAKE and that of OAKE. Specifically, when ruling out the forging attack, the input of $\mathcal{F}$ consists of only the value B, while the value $X_0 = g^{x_0}$ to be sent as the DH-component from $\hat{A}$ in the test-session is generated by $\mathcal{F}$ itself and thus x_0 is known to $\mathcal{F}$. Then, the assumed ability of forging attack is reduced to the rOAKE-HDR signature under merely the GDH assumption for this case of $\hat{A} = \hat{B}$ but $X_0 \neq Y_0$. Finally, the SK-security analysis of rOAKE, for the case of $\hat{A} = \hat{B}$ and $X_0 = Y_0$, is identical to that of OAKE. In particular, for this special case, all the pre-computed values $(x, X, B^{a+dx}$ and (y, Y, A^{b+cy}) can be exposed to the attacker for non-matching incomplete sessions. We have the following corollary:

Corollary G.8 *The rOAKE protocol (actually, the variant with public-keys removed from the inputs of c, d, e) is SK-secure in the random oracle model, under the following assumptions and secrecy exposure:*

- *The GDH assumption, with offline pre-computed and exposable DH-components and DH-exponents, for the case $\hat{A} \neq \hat{B}$ and the case $\hat{A} = \hat{B}$ but $X_0 \neq Y_0$.*

- *The CDH assumption, with offline pre-computed and exposable DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say A^{b+cy} and B^{a+dx}), for the case $\hat{A} = \hat{B}$ and $X_0 = Y_0$.*

Notes on some inherent security limitations. The reader should beware of some inherent security limitations for any one-round and two-round implicitly-authenticated DHKE protocols, e.g., the PFS vulnerability for any two-round implicitly-authenticated DHKE and the KCI vulnerability for any one-round DHKE (more details are referred to [42]). Even for the three-round versions of (s,r)OAKE and HMQV with explicit mutual authentications, there are also some inherent limitations. For example, the protocol responder may not be able to get deniability in a fair way, in case the malicious protocol initiator just aborts after receiving the second-round message. Also, both the three-round (s,r)OAKE and (H)MQV suffer from the cutting-last-message attack [46], etc. We remark that losing deniability fairness to protocol responder and lacking correct delivery guarantee of the last message are inherent to the protocol structure of (s,r)OAKE and (H)MQV and do not violate the definition of the SK-security in the CK-framework, which though can be easily remedied but at the price of ruining the performance advantages and/or adding additional system complexity.

H More on Security of (s,r)OAKE Beyond the CK-Framework

In this section, we make some further investigations on the security properties of (s,r)OAKE not captured by the CK-framework, which further strengthens the security guarantee of the (s,r)OAKE protocols. The first observation is: the security analysis of (s,r)OAKE in the CK-framework also implies that (s,r)OAKE is resistant to reflection attacks.

H.1 Security with Public Computations

The work of [44] considers a new attack scenario for key-exchange protocols with public computations, where it is convenient to split an entity (performing a run of KE-protocol) into two parts: a trusted authentication device, and an untrusted computing device. The authentication device enforces the confidentiality of the authentication data, while some computing operations required by the protocol are *publicly* carried out by the (possibly untrusted) computing device. This allows to use an authentication device with little computing power, and to make computing devices independent from users [44].

The work [44] gives some concrete applications that might be benefited from public computations: (1) Mobile phones include smart cards which store the user authentication data; the handsets themselves are the computing devices. (2) PCs (corresponding to the computing device) equipped with a crypto token (corresponding to the authentication device) have a lot more computing power than the token itself, but may be plagued by spyware or virus. For more details, the reader is referred to [44].

(H)MQV with public computations. With the computation of $\hat{B}$ as an example (the same holds for $\hat{A}$), a natural split of authentication computation and public computation is as follows [44]: The authentication device generates (y, Y), forwards Y to the computation device; After getting $(\hat{A}, X)$ from the computation device, the authentication device computes $s = y + eb$, where $e = h(Y, \hat{A})$, and then forwards s to the computation device; After getting s from the authentication device, the computation device computes $K_{\hat{B}} = (XA^d)^s$, and then the session-key, and then communicate with $\hat{A}$ with the session-key.

One key point is: as we assume the computation device may not be trustful, once the value s is leaked to an attacker (who may compromise the computation device), then the attacker can definitely impersonate $\hat{B}$ to $\hat{A}$ in any sessions. Note that, by only compromising the computation device, the attacker does not learn the DH-exponent y and the private-key b. This shows that (H)MQV does not well support deployment in the public computation model.

(s,r)OAKE with public computations. For applications in such scenarios, the natural split of authentication computation and public computation for (s,r)OAKE is as follows, with the computation of $\hat{B}$ as an example (the similar hold for $\hat{A}$): (1) The authentication device generates (y, Y) and possibly A^{fb+cy} (in case the authentication device has learnt the peer identity $\hat{A}$) where $c = 1$ for sOAKE or $c = h(\hat{A}, A, Y)$ for OAKE and rOAKE, and then forwards Y and possibly A^{fb+cy} to the computation device; (2) After getting X from the computation device, the authentication device computes $s = db + ey$, where $d = h(\hat{B}, B, Y)$ and $e = h(X, Y)$ for OAKE and rOAKE (resp., $d = 1$ and $e =$

$h(\hat{A}, A, \hat{B}, B, X, Y)$ for sOAKE), and then forwards s to the computation device; (3) After getting s from the authentication device, the computation device computes $K_{\hat{B}} = A^{cy}X^s$, and then the session-key, and then communicates with $\hat{A}$ with the session-key. *Note that $y, Y, c, d, A^{fb+cy}, db$ can be offline pre-computed by the authentication device, and the authentication device needs only online computing ey and s. Also, the computation device essentially needs to compute only one exponentiation X^s.*

Below, we make some discussions about the security of (s,r)OAKE in the public computation model.[7]

Discussion on security of sOAKE with public computations. We note that, under the DLP assumption, the knowledge of (A^y, s) of a session of sOAKE, learnt by the adversary by compromising the computation device, is essentially useless for the attacker to violate other sessions other than the matching session $(\hat{B}, \hat{A}, Y, X)$. The reason is that $s = b + ey$ for sOAKE, where $e = h(\hat{A}, A, \hat{B}, B, X, Y)$ commits to the whole session-tag. Thus, the value s cannot be used by the attacker to violate a non-matching session, unless it can compute y from A^y (and thus b from s) which however is infeasible by the DLP assumption.

Discussion on security of OAKE and rOAKE ((r)OAKE in short) with public computations. The knowledge (A^{fb+cy}, s) of a session of OAKE, where $s = db + ey$, $d = h(\hat{B}, B, Y)$ and $e = h(X, Y)$, is essentially useless under the DLP assumption for the attacker to violate other sessions other than sessions of the tag $(\hat{A}^*, A^*, \hat{B}, B, X, Y)$ where $(\hat{A}^*, A^*)$ may be different from $(\hat{A}, A)$. As the DH-component X is generated by uncorrupted players randomly and independently, it implies that with overwhelming probability the knowledge of (A^{cy}, s) can only help the attacker to violate the security of *at most one unexposed non-matching session.*

For example, consider that the attacker interacts concurrently with $\hat{A}$ (in the name of $\hat{B}$) and $\hat{B}$ (in the name of $\hat{A}^* \neq \hat{A}$ but of the same public-key A); the attacker faithfully relays the DH-components X and Y in the two sessions; in case the attacker learns both s and the private-key a of $\hat{A}$, then it can impersonate $\hat{B}$ to $\hat{A}$ in *the unique session in which $\hat{A}$ sends X.*

On the one hand, we suggest the is quite unreasonable to assume the attacker get the ability to expose both s and the secret-key of $\hat{A}$. On the other hand, we remark this weakness is at the price of supporting the advantageous post-ID computability offered by OAKE. Though this weakness can be trivially remedied (by putting $\hat{A}$ into d and $\hat{B}$ into c), but at the price of sacrificing the advantage of post-ID computability. Even with this (seemingly unreasonable) weakness in the public computation model for OAKE in mind, the potential damage caused is still much mitigated in comparison with that of (H)MQV in such scenarios.

H.2 Resistance to KCI, and Weak PFS

Recall that the security of DHKE protocols in the CK-framework is w.r.t. an unexposed test-session $(\hat{A}, \hat{B}, X_0, Y_0)$, where $\hat{A}$ and $\hat{B}$ are uncorrupted parties (which implies both the private-keys a, b are not exposed to the attacker) but the value Y_0 may be generated by the attacker impersonating $\hat{B}$ (in this case, the matching session does not exist). In this section, we consider the security damage caused by compromising static secret-keys of players, i.e., one or both of the secret-keys a, b of the test-session are exposed to the attacker.

Firstly, we note that if both the peer $\hat{B}$ (in the test-session) is corrupted and the value Y_0 is generated by the attacker itself, then no security can be guaranteed for the test-session within the CK-framework (as the attacker can now compute the session-key by itself). In this section, we mainly investigate the resistance against key-compromise impersonation (KCI) attacks, and perfect forward security (PFS). Roughly speaking, a key-compromise impersonation attack is deemed successful if the attacker, knowing the private key a of a party $\hat{A}$ (which of course allows the attacker to impersonate $\hat{A}$), is able to impersonate another *different* uncorrupted party $\hat{B} \neq \hat{A}$ (for which the attacker does not know the secret-key b) to $\hat{A}$. Note that for KCI attacks, the attacker still can generate the DH-component Y_0 for the test-session (without the matching session then). The PFS property says that the leakage of

[7] We note that some modifications to (s,r)OAKE may be needed to give a formal proof in the public computation model, in accordance with the work of [44]. Here, we stress that (s,r)OAKE, particularly sOAKE, very well supports the public-computation model even without such modifications.

the static secret-key of a party should not compromise the security of session-keys ever established by that party, and erased from memory before the leakage occurred.

Definition H.1 (clean session [42]) *We say that a complete session of a key-exchange protocol is clean, if the attacker did not have access to the session's state at the time of session establishment (i.e., before the session is complete), nor it issued a session-key query against the session after completion.*

Note that, for a *clean* session at an uncorrupted party, the attacker did not issue a state-reveal query while the session was incomplete or a session-key query after completion. Moreover, the attacker was not actively controlling or impersonating the party during the session establishment (neither by making any choices on behalf of that party in that session or eavesdropping into the session's state).

Definition H.2 *[42] We say that a KE-attacker $\mathscr{A}$ that has learned the static secret-key of $\hat{A}$ succeeds in a KCI attack against $\hat{A}$, if $\mathscr{A}$ is able to distinguish from random the session-key of a complete session at $\hat{A}$ for which the session peer $\hat{B} \neq \hat{A}$ is uncorrupted (which implies the private-key of $\hat{B}$ is not exposed to $\mathscr{A}$) and the session and its matching session (if it exists) are clean.*

In other words, the definition says that, as long as the attacker is not actively controlling or observing the secret choices (particularly the ephemeral DH-exponent x) of the test-session, then even the knowledge of $\hat{A}$'s private-key still does not allow $\mathscr{A}$ to compromise the session-key. In particular, in such a protocol $\mathscr{A}$ cannot impersonate an uncorrupted party $\hat{B}$ to $\hat{A}$ in a way that allows $\mathscr{A}$ to learn any information about the resultant session-key [42] (even if the attacker impersonates $\hat{B}$ and generates the DH-component, say Y_0, by itself).

Proposition H.1 *Under the GDH assumption in the random oracle model, the (s,r)OAKE protocols (actually, their public-key free variants), with offline pre-computation, resist KCI attacks in the CK-framework.*

The resistance of (s,r)OAKE to KCI attacks is essentially implied by the proofs of Theorem G.1, Theorem G.2 and Corollary G.8 for the case of $\hat{A} \neq \hat{B}$, from the observations that: for KCI attacks the test-session is between a pair of different uncorrupted peers $\hat{A} \neq \hat{B}$, and the security of the underlying (s,r)OAKE-HDR holds even if the forger learns the private-key of the uncorrupted peer (the party $\hat{A}$ here).

Weak PFS (wPFS). It is clarified in [42] that, no 2-round DHKE protocols *with implicit key confirmation* can fully render PFS security (the 3-round versions of HMQV and (s,r)OAKE, with explicit key-confirmation and mutual authentications, do fully provide PFS property). The work [42] formulates a weak notion of PFS, named weak PFS (wPFS), and shows that HMQV satisfies this wPFS property. Roughly speaking, wPFS property says that *if the attacker is not actively involved with the choices of X, Y at a session* (particularly if it does not get to choose or learn the DH-exponent x or y), then the resultant session-key does enjoy forward security. Formally,

Definition H.3 *[42] A key-exchange protocol provides wPFS, if an attacker $\mathscr{A}$ cannot distinguish from random the key of any clean session $(\hat{A}, \hat{B}, X, Y)$, where Y is also generated by an uncorrupted party in a clean session, even if $\mathscr{A}$ has learned the private keys of both $\hat{A}$ and $\hat{B}$.*

Proposition H.2 *Under the CDH assumption, the (s,r)OAKE protocols provide wPFS property in the random oracle model.*

For establishing the wPFS property for (s,r)OAKE, we do not need here to construct a (s,r)OAKE-HDR forger from the attacker violating the wPFS property. Actually, we can directly reduce the loss of wPFS to the CDH assumption, from the following observations: given the knowledge of both a and b, the computation of $K_{\hat{A}}$ or $K_{\hat{B}}$ is reduced to the computation of g^{xye} from the random DH-components X, Y. Recall that, for wPFS property, we assume the attacker is not actively involved with the choices of X, Y. Then, we can simply guess the test-session, and set the DH-components as some given random elements X, Y, and then reduce the ability of the attacker to violate wPFS *directly* to the CDH assumption. More details are omitted here.

I More on Computational Fairness

We first present more discussions on the formulation of session-key computational fairness.

We note that the issue of computational fairness can apply to interactive protocols in general, as long as the honest players have the same computational operations under protocol specifications.[8] For implicitly authenticated DHKE protocols like (H)MQV and OAKE, we only considered here the session-key computational fairness. In general, for key-exchange protocols with explicit authentication (e.g., via signatures and/or MACs), besides session-key computational fairness, we need also consider authentication computational fairness. The formulation of session-key computational fairness is also instrumental in formulating authentication computational fairness, which is beyond the scope of this work.

Computational fairness vs. contributiveness. A related notion, called *contributiveness*, was also introduced in the literature of *group* key-exchange (see e.g., [52, 11]. Roughly speaking, the notion of contributiveness for group key-exchange says that a subset of players cannot pre-determine the session-key output. But, contributiveness says nothing about computational fairness in computing the session-key output. As clarified in Section 5, computational fairness says that each player needs to compute the same number of *non-malleably independent* dominant-operation values in generating the session-key output. To our knowledge, the notion of non-malleably independent dominant-operation values was not previously considered in the literature. If we view each non-malleably independent dominant-operation value as a proof-of-knowledge of the corresponding secrecy, our notion of computational fairness ensures that a subset of malicious players cannot set the session-output to be some value that can be publicly computed from the session transcript. From these observations, we can see that computational fairness and contributiveness are two fundamentally different notions. Based on this work, computationally-fair group key-exchange is investigated in another separate work.

Dominant operation values for OAKE and (H)MQV. Before proceeding to analyze the computational fairness of OAKE and (H)MQV, we first specify the underlying dominant operation values. Recall that it is the task of the protocol designer to specify the dominant operation values, with respect to which computational fairness will be provably proved. For OAKE and any complete session-tag $Tag = (\hat{A}, \hat{B}, X = g^x, Y = g^y)$, the dominant operation values specified for the player $\hat{A}$ (resp., $\hat{B}$) are $V_1^{\hat{A}} = B^{dx} \in G \setminus 1_G$ and $V_2^{\hat{A}} = Y^{ca+ex} \in G$ (resp., $V_1^{\hat{B}} = A^{cy}$ and $V_2^{\hat{B}} = X^{db+ey}$), where $c = h(\hat{A}, A, Y), d = h(\hat{B}, B, X), e = h(X, Y)$. For (H)MQV, as they were not designed for computational fairness, dominant operation values were unspecified. We consider two natural specifications of dominant operation values for (H)MQV:

- Specification-1: For any complete session-tag $Tag = (\hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for the player $\hat{A}$ (resp., $\hat{B}$) are $V_1^{\hat{A}} = YB^e \in G$ and $V_2^{\hat{A}} = x + da \in Z_q$ (resp., $V_1^{\hat{B}} = XA^d \in G$ and $V_2^{\hat{B}} = y + eb \in Z_q$), where $d = h(X, \hat{B}), e = h(Y, \hat{A})$ for HMQV (resp., $d = 2^l + (X \mod 2^l)$ and $e = 2^l + (Y \mod 2^l)$ for MQV). The key derivation function F_K, in this case, is specified to be $F_K(V_1, V_2, Tag) = H_K(V_1^{V_2})$ for both OAKE and (H)MQV.
- Specification-2: For any complete session-tag $Tag = (\hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for the player $\hat{A}$ (resp., $\hat{B}$) are $V_1^{\hat{A}} = Y^{x+da} \in G$ and $V_2^{\hat{A}} = B^{e(x+da)} \in G$ (resp., $V_1^{\hat{B}} = X^{y+eb}$ and $V_2^{\hat{B}} = A^{d(y+eb)}$), where $d = h(X, \hat{B}), e = h(Y, \hat{A})$ for HMQV (resp., $d = 2^l + (X \mod 2^l)$ and $e = 2^l + (Y \mod 2^l)$ for MQV). The key derivation function F_K, in this case, is specified to be $F_K(V_1, V_2, Tag) = H_K(V_1 \cdot V_2)$ for both OAKE and (H)MQV.

Proposition I.1 *In the random oracle model where the hash function $h : \{0,1\}^* \to Z_q^*$ is assumed to be a random oracle, OAKE is session-key computationally fair assuming, while (H)MQV is not, with respect to the above specified dominant operations.*

[8] In particular, most key-exchange protocols are protocols of such type, while key distribution protocols (e.g., via public-key encryption) are not.

Proof. For OAKE, we show that the distribution of (V_1^I, V_2^I), for both $I \in \{\hat{A}, \hat{B}\}$, is statistically indistinguishable from that of (U_1, U_2) in the random oracle model, where U_1 (resp., U_2) is taken uniformly at random from $G \setminus 1_G$ (resp., G). Clearly, V_1^I distributed uniformly over $G \setminus 1_G$ assuming $h : \{0,1\}^* \to Z_q^*$ to be a random oracle. But, the distribution of V_2^I is not identical to the uniform distribution of U_2 over G. Specifically, in the random oracle model, $\Pr[V_2^I = 1_G] = \frac{1}{q-1}$, while for any $\alpha \in G \setminus 1_G$ $\Pr[V_2^I = \alpha] = \frac{q-2}{(q-1)^2}$. But, by straightforward calculation, we have that the statistical distance between V_2^I and U_2 is $\frac{1}{q(q-1)}$, which is negligible in $l = |q|$. That is, (V_1^I, V_2^I) are *statistically* non-malleably independent. Thus, for OAKE, the session-key computation by each(whether honest or not) user involves the same number, say two, statistically non-malleably independent dominant operation values, and computational fairness property follows.

For (H)MQV, our concrete EDA attacks presented in Section 5 demonstrate that both MQV and HMQV do not satisfy computational fairness, with respect to either the above Specification-1 or Specification-2 of dominant operation values. Specifically, consider the following specific relations (corresponding to the two specific cases of our attack):

- For Specification-1, $R(V_1, V_2, Tag) = 1$ iff $V_1^{V_2} = 1_G$, or, $R(V_1, V_2, Tag) = 1$ iff $V_1^{V_2} = YB^e$, where $V_1 \in G$, $V_2 \in Z_q$ and YB^e can be publicly computed from the session-tag Tag.
- For Specification-2, $R(V_1, V_2, Tag) = 1$ iff $V_1 \cdot V_2 = 1_G$, or, $R(V_1, V_2, Tag) = 1$ iff $V_1 \cdot V_2 = YB^e$, where $V_1 \in G$, $V_2 \in G$ and YB^e can be publicly computed from the session-tag Tag.

For all these specific relations, there exist complete session-tags Tag (corresponding to the sessions caused by the EDA attacks presented in Section 5) such that $\Pr[R(V_1, V_2, Tag) = 1] = 1$, while $\Pr[R(U_1, U_2, Tag) = 1] = 1$ is always negligible where (U_1, U_2) are taken uniformly at random from$G \times Z_q$ (resp., $G \times G$) for Specification-1 (resp., Specification-2). □

Remark: By the session-key computational fairness property of OAKE, the session-key computation involves two non-malleably independent values A^{cy} and X^{db+ey} no matter how a malicious $\hat{B}$ does (i.e., $\hat{B}$ is infeasible to make the values A^{cy} and X^{db+ey} correlated under any predetermined polynomial-time computable relation). If we view each non-malleably independent exponentiation value as a proof-of-knowledge of the corresponding exponent, then to compute the session-key any PPT player has to "know" both cy and $db + ey$, from which both the static secret-key b and the ephemeral DH-exponent y can be efficiently derived. In this sense, the session-key computation of OAKE itself can be viewed as a non-malleable join proof-of-knowledge of both b and y. This further implies that a malicious player is infeasible to set the session-key to some values that can be publicly computed from the session transcript.

On fixing HMQV to achieve computational fairness. In [67, 66], we proposed some variants of (H)MQV, *just in the spirit of (s,r)OAKE and NMJPOK* to prevent our EDA attacks and to render the property of session-key computational fairness. The key point is to put A (resp., B) into the input of d (resp., e). Specifically, we have the following fixing approaches, by setting (1) $d = h(X, \hat{B}, A)$ and $e = h(Y, \hat{A}, B)$; or (2) $d = h(\hat{A}, A, \hat{B}, B, X, Y)$ and $e = h(d)$; or (3) $d = h(\hat{A}, A, X)$ and $e = h(\hat{B}, B, Y)$, etc. Other components remain unchanged. For the above third fixing solution, in order to get only one exponentiation online efficiency, we can make some further modifications by setting $K_{\hat{A}} = (Y^e B)^{xd+a}$, $K_{\hat{B}} = (X^d A)^{ye+b}$, where $d = h(\hat{A}, A, X)$ and $e = h(\hat{B}, B, Y)$. The session-key is still $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$. For presentation simplicity, we refer to this solution as the fourth fixing solution (this protocol variant is named as OAKE-MQV in [67, 66]).

Unfortunately, we failed in providing the provable security for any of the above HMQV variants in the CK-framework. In particular, we do not know how to extend the security proof of HMQV in [42] to any of the above four fixing solutions. Indeed, HMQV was very carefully designed to enjoy provable security in the CK-framework, and the HMQV structure is quite sensitive to the security analysis in the CK-framework. Besides lacking provable security in the CK-framework, many other advantageous features enjoyed by (s,r)OAKE (as clarified in Section 4) are also lost with the above fixing solutions. The surrounding issues are quite subtle and tricky, and indeed (s,r)OAKE was very carefully designed to achieve all these advantageous features (with a new protocol structure, some new design rationales and building tools, and also inspired by the analysis of HMQV and the design of deniable IKE [68]).