A Complementary Analysis of the (s)YZ and DIKE Protocols

Augustin P. Sarr[1], Philippe Elbaz–Vincent[2], and Jean–Claude Bajard[3]

[1]Université de Strasbourg <aug.sarr@gmail.com>
[2]Institut Fourier – CNRS, Université Grenoble 1
[3]LIP6 – CNRS, Université Paris 6

Abstract

The Canetti–Krawczyk (CK) model remains widely used for the analysis of key agreement protocols. We recall the CK model, and its variant used for the analysis of the HMQV protocol, the $\mathrm{CK}_{\mathrm{HMQV}}$ model; we recall also some of the limitations of these models. Next, we show that the (s)YZ protocols do not achieve their claimed $\mathrm{CK}_{\mathrm{HMQV}}$ security. Furthermore, we show that they do not achieve their claimed computational fairness. Our attack suggests that no two–pass key exchange can achieve this attribute. We show also that the Deniable Internet Key Exchange fails in authentication; this illustrates the inability of capturing some impersonation attacks in the CK model. Besides, we propose a secure, efficient, and deniable protocol, geared to the post peer specified model.

Keywords: key exchange, (e)CK models, (s)YZ, DIKE, computational fairness, SMQV–1.

1 Introduction

The design and analysis of key agreement protocols is a notoriously subtle topic; and a large part of the proposed designs appears to be flawed. In [1], Bellare and Rogaway proposed a new approach for the analysis of key agreement protocols. Since their work, other models was proposed, including the Canetti–Krawczyk (CK) [3] and the extended Canetti–Krawczyk (eCK) [11] models, which are now considered as advanced security definitions. However, there remains a large class of attacks which are not considered in these security definitions [6, 18].

Recently, Yao and Zhao [21] proposed a new family of authenticated key exchange protocols, the (s)YZ protocols, they analyzed in the $\mathrm{CK}_{\mathrm{HMQV}}$ model (the variant of the CK model used in [10] for the analysis of the HMQV protocol). They introduced also a new security notion, *computational unfairness*, which aims to capture the difference that may exist between the computational effort it requires for an honest and a malicious party to complete a session; and showed that the (s)YZ protocols are computationally fair, while (H)MQV is not. Also, in [23], they proposed a new protocol termed Deniable Internet Key Exchange (DIKE), which "adds novelty and new value to the IKE" standard; they show the DIKE secure in the CK model.

In this work, we propose a complementary analysis of the (s)YZ and DIKE protocols; we re–examine also the computational unfairness security attribute. And, we show that, contrary to the claims of [21], the (s)YZ protocols do not achieve the $\mathrm{CK}_{\mathrm{HMQV}}$–security. We show also that the (s)ZY protocols do not the achieve the computational fairness attribute, our attack suggests that no two–pass key exchange protocol can achieve this attribute. Besides, we show that the DIKE protocol fails in authentication. Namely, if an attacker learns the ephemeral secret at a session initiator, it can impersonate any party to the initiator; this illustrates the inability of capturing some impersonation attacks in the CK model.

The remainder of this paper is organized as follows. In section 2, we recall the CK and $\mathrm{CK}_{\mathrm{HMQV}}$ security models, and some of their limitations. In section 3, we recall the YZ protocols family. In section 4 we discuss the YZ protocols attributes, an propose an attack which invalidates their $\mathrm{CK}_{\mathrm{HMQV}}$ security, we also show that these protocols do not achieve their claimed computational fairness. In section 5, we recall the DIKE protocol, and propose an attack which shows its failure in authentication, we also present a SMQV variant geared to the post peer specified model. We conclude in section 6. In the appendix we propose a reductionist security proof of a variant of the FXCR–1 scheme [18], to clarify the misunderstanding from [24] about the SMQV security arguments.

Notations and terminology: $\mathcal{G}$ denotes a multiplicatively written cyclic group of prime order q generated by G, $|q|$ is the bit length of q; $\mathcal{G}^*$ is the set of non–identity elements in $\mathcal{G}$. For $X \in \mathcal{G}$, the lowercase x denotes the discrete logarithm of X in base G. The identity of a party with public key A is denoted $\hat{A}$; $\hat{A}$ is supposed to contain A, and may be a certificate. If $\hat{A} \neq \hat{B}$, we suppose that no substring of $\hat{A}$ equals $\hat{B}$; H and H_K are λ–bit hash function, where λ is the length of session keys, and $\bar{H}$ is a l–bit hash function, where $l = (\lfloor \log_2 q \rfloor + 1)/2$. The symbol $\in_R$ stands for "chosen uniformly at random in."

2 The Canetti–Krawczyk Security Model(s)

In the CK model, a protocol is defined as a collection of procedures run by a finite number of parties. Each protocol specifies its processing rules for incoming and outgoing messages. A *key exchange* is a protocol which involves two parties. A *session* is an instance of a protocol run at a party. In a key exchange, each session is activated with a quadruple $(\hat{P}_i, \hat{P}_j, \psi, \varsigma)$, where $\hat{P}_i$ is the session owner, $\hat{P}_j$ is the peer, ψ is the session identifier, and ς is the role of $\hat{P}_i$ in the session. A session identifier is required to be unique at each party involved in the session, i.e., a party never uses the same identifier twice. Two sessions with activation parameters $(\hat{P}_i, \hat{P}_j, \psi, \varsigma)$ and $(\hat{P}_j, \hat{P}_i, \psi', \varsigma')$ are said to be matching if $\psi = \psi'$. Notice that matching CK–sessions can have the same role.

The adversary considered in this model, denoted here by $\mathcal{A}$, is a probabilistic polynomial time machine in control of communications between parties; outgoing messages are submitted to $\mathcal{A}$, which decides about their delivery. The adversary decides also about session activations. In addition, it is given the following queries, aiming to model practical information leakages.

- *SessionStateReveal*$(\hat{P}_i, \psi)$. When this query is issued on the uncompleted session ψ at $\hat{P}_i$, the adversary obtains the ephemeral information contained in the session. However, the model does not specify the information revealed by this query; it leaves this to be specified by protocol designers.
- *SessionKeyReveal*$(\hat{P}_i, \psi)$. With this query, the adversary obtains, the session key derived in the session ψ at $\hat{P}_i$ if it is completed and unexpired.
- *Corrupt*$(\hat{P}_i)$. When this query is issued on a party $\hat{P}_i$, the adversary obtains all the information the party holds, including its static private key and session states. Once the query is issued, the attacker (which is in control of communication links) can impersonate the party at will; one then consider the party under the attacker's control. A party against which this query is not issued is said to be *honest*.
- *Expire*$(\hat{P}_i, \psi)$. This query models the erasure of a session key and state from the session owner's memory. Notice that a session can be expired while its matching session is unexpired.
- *Test*$(\hat{P}_i, \psi)$. When the this query is issued on a completed (and unexpired) session ψ at $\hat{P}_i$, a bit γ is chosen at random, and depending on the value of γ, the attacker is provided with either the session key, or a random value chosen under the distribution of session keys. The attacker is allowed to continue its run with regular queries, but not to reveal the test session or its matching session's key or state.

Definition 1 (CK Session Freshness)**.** A session at a party is said to be *locally exposed* if it was sent a *SessionStateReveal* query, a *SessionKeyReveal* query, or if its owner is corrupted. A session is said to be *exposed* if it or its matching session is locally exposed. An unexposed session is said to be *CK–fresh*.

With this session freshness definition, a CK–secure protocol is as follows.

Definition 2 (CK–Security)**.** A protocol is said to be *CK–secure* if:
(1) when two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key, and
(2) no polynomially bounded adversary can distinguish a *CK–fresh session* key from a random value (chosen under the distribution of session keys) with probability significantly greater than 1/2.

2.1 The CK$_{\mathbf{HMQV}}$ Model

In [10], Krawczyk uses a variant of the CK–model for the analysis of the HMQV protocol. In this variant, termed here CK$_{\text{HMQV}}$, a session is identified with a quadruple $(\hat{P}_i, \hat{P}_j, X, Y)$ where $\hat{P}_i$ is the session owner, $\hat{P}_j$ is the peer, and X and Y are respectively the outgoing and incoming ephemeral public keys. Two sessions with identifiers $(\hat{P}_i, \hat{P}_j, X, Y)$ and $(\hat{P}_j, \hat{P}_i, Y, X)$ are said to be *matching*. In addition, there is no *Expire* query in the CK$_{\text{HMQV}}$ model (the motivation is that the CK_{HMQV} model does not aim to capture *forward secrecy*[1], but *weak forward secrecy*).

In separate analyses, Krawczyk [10, sections 6 and 7.4] allows the adversary to query a *SessionStateReveal* on the test session, or to learn the static private key of the test–session owner. Notice that, the purpose of [10], was not to propose a new security model, as it refers to [3] for details [10, p. 9], and considers its session identifiers and matching sessions definitions as consistent with the CK–model [10, p. 10], while the CK and CK_{HMQV} models are formally and practically incomparable [5, 6]. Except the differences on (1) session identifiers and matching sessions definition, and (2) the absence of the *Expire* query in the CK_{HMQV} model, the CK and CK_{HMQV} session freshness and security definitions are the same.

Some Weaknesses in the $\text{CK}_{(\text{HMQV})}$ Models

In this section, we recall some of the reported weaknesses on the CK and CK_{HMQV} security models. The discussion is voluntarily limited to the aspects related to this work. Notice that contrary to [21], and in accordance with [2, 4, 5, 18, 6], by CK–model we mean the security model defined in [3], and by CK_{HMQV}, the model(s) used in [10] for the analysis of the HMQV protocol.

On the CK Matching Sessions. Besides not modelling *key compromise impersonation resilience,* a main limitation of the CK model is its matching sessions definition. Indeed, it is a requirement of the model that the layer calling a key exchange makes sure that a party never uses the same identifier twice. One may ask how can this be achieved over an unauthenticated network in control of an active adversary.

Moreover, in the CK model, session identifiers may be nonces generated by the session initiator and provided to the peer in the first message of the protocol. In this case, when each party stores the previously used identifiers, and verifies at session activation that the identifier was not used before; the requirement that a party never uses the same identifier twice is then achieved. Unfortunately, in such a case, the model fails in capturing some practical impersonation attacks. A kind of impersonation attacks not captured in the CK model, was reported and illustrated in [18, section 2] on the protocol $\mathcal{P}$ from [15].

Limitation of the CK_{HMQV} Session Identifiers. By defining matching sessions using the identities and ephemeral keys of the involved parties, the CK_{HMQV} model, patches the impossibility of capturing, in the CK model, the kind of impersonation attacks reported in [18, sect. 2]. However, the CK_{HMQV} session identifiers and matching sessions definitions, remain very limited. Indeed, as reported in [5, 6], protocols such that the way a party involved in a session derives the session key is dependant from its role (*initiator* or *responder*) are insecure in the CK_{HMQV} model. The security of such protocols can be invalided by an attacker performing as follows.

[1]Some authors, [10] for instance, use the term ‘perfect forward secrecy’, but following [2], we prefer ‘forward secrecy’ to avoid a confusion with Shannon’s ‘perfect secrecy’.

Attack 1 Crossing Attack to invalidate $\mathrm{CK}_{\mathrm{HMQV}}$ security

```
(1) Activate a session at $\hat{A}$ with peer $\hat{B}$, and intercept $\hat{A}$'s message to $\hat{B}$.
(2) Activate a session at $\hat{B}$ with peer $\hat{A}$, and intercept $\hat{A}$'s message to $\hat{B}$.
(3) Send $\hat{B}$'s message to $\hat{A}$ as a responder in the session initiated by $\hat{A}$.
(4) Send $\hat{A}$'s message to $\hat{A}$ as a responder in the session initiated by $\hat{B}$.
```

In this attack, the sessions at $\hat{A}$ and $\hat{B}$ are $\mathrm{CK}_{\mathrm{HMQV}}$ matching; but, when the key derivation is role dependant, they do not yield the same session key. As a result, the first requirement in the $\mathrm{CK}_{\mathrm{HMQV}}$ security definition is not satisfied. Such attacks invalidate a large class of valuable protocols. Consider, for instance, the HMQV protocol. Recall that in an HMQV execution, the initiator, say $\hat{A}$, chooses and ephemeral key $X = G^x$ ans sends $(\hat{A}, \hat{B}, X)$ to the responder $\hat{B}$; $\hat{B}$ chooses an ephemeral key $Y = G^y$ and sends $(\hat{B}, \hat{A}, Y)$ to $\hat{A}$. Both $\hat{A}$ and $\hat{B}$ compute the dual signature $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$, wherein $d = \bar{H}(X, \hat{B})$ and $e = \bar{H}(Y, \hat{A})$; and the session key $K = H(\sigma)$. One can see, through Attack 1 that the HMQV variant wherein the session key is derived as $H(\sigma, X, Y)$ instead of $K = H(\sigma)$ is insecure in the $\mathrm{CK}_{\mathrm{HMQV}}$ model; the reason is that matching CK sessions do not necessarily yield the same key. Indeed, when an attacker performs as in Attack 1, $\hat{A}$ derives $K_A = H(\sigma, X, Y)$, while $\hat{B}$ derives $K_B = H(\sigma, Y, X) \neq K_A$; they do not compute the same session key. The $\mathrm{CK}_{\mathrm{HMQV}}$ matching sessions definition formally invalidates a large class of valuable designs.

3 The YZ Protocols Family

In [22] Yoa and Zhao introduce a family of authenticated Diffie–Hellman protocols. The building blocks of the protocols, termed *joint proof of knowledge*, can be viewed as variants of the (D)XCR scheme. The general construction of the secret goup element, shared between two parties, say $\hat{A}$ and $\hat{B}$ with respective ephemeral keys X and Y is $Z = B^{fa+dx}Y^{ca+ex} = A^{fb+cy}X^{db+ey}$, where c, d, e and f are publicly computable digest values. The main instantiations of the protocols, YZ and sYZ protocols, are close enough. An execution of the YZ protocol is as in Protocol 2.

To achieve *reasonable deniability*, the inability of an attacker to compute a session key from ephemeral private keys is sacrificed. Notice also, that the inclusion of the static keys in the messages is superfluous, as in effect $\hat{A}$ and $\hat{B}$ are certificates.

To obtain the *single–hash YZ* (sYZ) variant, it suffices to modify Protocol 2, with the digests d, c and e set to $d = c = 1$ and $e = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$. When $c = \tilde{H}(\hat{A}, A, \hat{B}, B, Y)$, $d = \tilde{H}(\hat{B}, B, \hat{A}, A, X)$, $e = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$, and $f = \tilde{H}(\hat{A}, A, \hat{B}, B)$, the resulting protocol is termed *robust YZ* (rYZ); and *single hash robust YZ* (srYZ) when $c = d = f = 1$, and $e = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$.
In [24], other protocols, inspired by the (H)MQV protocols are proposed, namely the YZ–MQV–i $_{i=1,4}$ protocols, defined with the following digest values.

- YZ–MQV–1: $c = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$, $d = \tilde{H}(c), e = 1$, and $f = cd$.
- YZ–MQV–2: $d = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$, $c = \tilde{H}(d), e = cd$, and $f = 1$.
- YZ–MQV–3: $c = \tilde{H}(\hat{A}, A, X, pub_1)$, $d = \tilde{H}(\hat{B}, B, Y, pub_2)$, $e = 1$, and $f = cd$.

Protocol 2 The YZ Protocol

I) The initiator $\hat{A}$ does the following:
   (a) Choose $x \in_R [1, q-1]$ and compute $X = G^x$.
   (b) Send $(\hat{A}, A, X)$ to the peer $\hat{B}$.

II) At receipt of $(\hat{A}, A, X)$, $\hat{B}$ does the following:
   (a) Verify that $X \in \mathcal{G}^*$.
   (b) Choose $y \in_R [1, q-1]$ and compute $Y = G^y$.
   (c) Send $(\hat{B}, B, Y)$ to $\hat{A}$.
   (d) Compute $c = \tilde{H}(\hat{A}, A, Y)$, $d = \tilde{H}(\hat{B}, B, X)$, and $e = \tilde{H}(Y, X)$.
   (e) Compute $\sigma = A^{cy} X^{db+ey}$ and $K = H(\sigma)$.

III) At receipt of $(\hat{B}, B, Y)$, $\hat{A}$ does the following:
   (a) Verify that $Y \in \mathcal{G}^*$.
   (b) Compute $c = \tilde{H}(\hat{A}, A, Y)$, $d = \tilde{H}(\hat{B}, B, X)$, and $e = \tilde{H}(Y, X)$.
   (c) Compute $\sigma = B^{dx} Y^{ca+ex}$ and $K = H(\sigma)$.

IV) The shared session key is K.

- YZ–MQV–4: $c = \tilde{H}(\hat{B}, B, Y, pub_2)$, $d = \tilde{H}(\hat{A}, A, X, pub_2)$, $e = cd$, and $f = 1$.

Contrary to the (s)YZ protocols, these protocols do not sacrifice the inability of an attacker to compute a session key from ephemeral private keys, and are not reasonably deniable.

On the Origins of the YZ–MQV Protocols. In [24, p. 352] one can read that the YZ–MQV–4 protocol is named YZ–MQV in [20] (the prefix in the protocol names have changed from 'YYZ' to 'YZ'), and that the SMQV and FHMQV protocols are originated from [20]. The authors of [24] refer also to the Chinese version of [20], but as we do not read Chinese we use [20].

The claim that YZ–MQV–4 protocol is named YZ–MQV in [20], and that SMQV and FHMQV are originated from [20] is erroneous. Indeed, there is a fundamental difference between the YZ–MQV–4 protocol and the YZ–MQV protocol, defined in [20, Claim 30] and in [21, p. 49], also drawn in [20, Figure 2]. In the YZ–MQV protocol, the values of c, d, e, and f are defined as $c = \tilde{H}(\hat{A}, A, X)$, $d = \tilde{H}(\hat{B}, B, Y)$, $e = 1$, and $f = cd$. This protocol increases the HMQV weakness pointed in [18], and fixed in the FHMQV and SMQV designs. In the YZ–MQV protocol, when an attacker learns an intermediate secret exponent in a session, it can not only *impersonate indefinitely* the session owner to its peer in the leaked session, as in HMQV [18], but to *any party*.

The other protocols defined in [20] do not match the YZ–MQV–$i_{i=3,4}$ protocols. One can find definitions parametrized by "public information" $pub_i, i = 1, \cdots, 4$ in [20, claims 24–25] where pub_i comprises a subset of $\{\hat{A}, A, m_1, \hat{B}, B, m_2\}$, but none of them matches YZ–MQV–$i_{\,i=3,4}$, as m_1, m_2 "denote other messages, other than the DH–components, exchanged in (or related to) the protocol run" [20, p. 18].

Even if the FHMQV and SMQV protocols can be instantiated from the YZ–MQV–$i_{\,i=3,4}$ protocols, which appeared after the FHMQV and SMQV protocols was proposed, we cannot see how SMQV and FHMQV can be considered as originated from [20]. Also, the description of the rYZ protocol have changed between [20] and [24]; in [20, pp. 20–21], the two possible values of f are 0 and 1, while in [24] $f = \tilde{H}(\hat{A}, A, \hat{B}, B)$.

4 On the Attributes of the (s)YZ Protocols

In this section, we discuss the (in)validity of the main attributes of the (s)YZ protocols. Namely, we show that the their claimed security and *computational fairness* do not hold. Before, we show that the protocols efficiency also should be temperated.

Temperating the (s)YZ Protocols Efficiency

The YZ protocols are reasonably efficient. They are are presented as *optimally efficient* [22, 24] meaning that no protocol achieving the same security attributes can be more efficient; but there is no proof of such a claim. The "optimal efficiency" of the (s)YZ protocols is sustained by the ability of a party, say $\hat{B}$, to precompute A^{fb+cy}, where A is the initiator's static key. Unfortunately, this argument is not careful enough. In fact, in an execution of a protocol from YZ family, the ability of $\hat{B}$ to precompute A^{Fb+Cy} may be very limited. The reason is that $\hat{B}$ usually evolves in an open network, often with a large number of potential peers; so knowing in advance the next peer seems difficult. This is particularly true, if $\hat{B}$ is a server with a large number of clients (a bank server, for instance). In this case, pre–computability is highly desirable, but the server cannot guess the next client; as a result, it is difficult to see how the A_i^{fb+cy}'s can be adequately precomputed.

4.1 The (s)YZ Protocols (In)Security

In [22] the (s)YZ protocols are shown secure in the $\mathrm{CK_{HMQV}}$ model under the Gap Diffie–Hellman assumption. As in these protocols, the session key derivation is role dependant, it is simple to show that matching sessions do not necessarily yield the same key; hence the $\mathrm{CK_{HMQV}}$ security arguments for the (s)YZ protocols are invalid. The security arguments are simply invalidated by Attack 1. When an attacker executes Attack 1, the two parties involved in the run, say $\hat{A}$ and $\hat{B}$, both believe being the session initiator, and respectively compute $e_A = \tilde{H}(X, Y)$ and $e_B = \tilde{H}(Y, X) \neq e_A$ (except with negligible probability). So the initiator and responder do not compute the same secret group element, and they do not derive the same session key, while their sessions are matching. The requirement that if two honest parties complete matching session, they should compute the same session key is not achieved, the (s)YZ protocols do not achieve $\mathrm{CK_{HMQV}}$ security.

Notice that, unless changed fundamentally, using the CK session identifiers and matching sessions definition does not patch the security arguments invalidity. As in the (s)YZ protocols, session keys can be computed from ephemeral private keys, it is also simple to show that these protocols are insecure the (s)eCK security models. Also, as these protocols sacrifice the inability of an attacker to compute a session key from ephemeral secrets, they cannot meet the principle that an attacker should not be able to impersonate a party, unless it knows its static private key. This is illustrated in section 5 against the DIKE protocol; it is not difficult to see that the attack holds against the (s)ZY protocols also.

In [24, Remark 2 or Table 2], the srYZ protocol is presented as seCK–secure. Here also, this is erroneous. As in the srYZ protocol $c = d = f = 1$, and $e = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$ [24, p. 351], the shared secret group element in an execution of this protocol is

$$\sigma = B^{a+x}Y^{a+ex} = A^{b+y}X^{b+ey};$$

Attack 3 invalidates the protocol's security in the seCK model.

Attack 3 Attack against srYZ in the seCK model

```
(1) Choose $x \in_R [1, q-1]$, and send $(\hat{A}, A, X' = g^x/A)$ to $\hat{B}$.
(2) Intercept $\hat{B}$'s response to $\hat{A}$ $(\hat{B}, B, Y)$.
(3) Issue and *EphemeralKeyReveal* query on the session at $\hat{B}$ to learn $y$.
(4) Use $x$ and $y$ to compute the same session key as $\hat{B}$, and communicate with
    $\hat{B}$ on behalf of $\hat{A}$.
```

Using x and y, the attacker can compute the same key as $\hat{B}$, for the secret $\hat{B}$ derives is

$$Z_B = A^{b+y}X'^{b+ey} = A^{b+y}(g^x/A)^{b+ey} = A^{(1-e)y}(BY^e)^x.$$

With Attack 1, we show that the (s)YZ protocols do not achieve $\mathrm{CK}_{\mathrm{HMQV}}$ security, the protocols are also eCK insecure; and Attack 3 invalidates the srYZ protocol's security in the seCK model.

Computational (Un)fairness

In [22], Yoa and Zhao introduce a new notion, termed *computational unfairness* between an attacker and an honest player. The aim is to capture the ability of an adversary to compute a secret shared with an honest party, while performing much less operations than the honest user does. Such attacks can be used, in turn, to mount effective Denial of Service (DoS) attacks. For concreteness, we recall in Attack 4 the *exponent–dependent attack* (EDA) proposed in [21, 22] to show that the (H)MQV protocols are computationally unfair. Recall that besides the differences on the digest values, computed as $d = 2^l + X \mod 2^l$ and $e = 2^l + Y \mod 2^l$ in MQV, the MQV and HMQV descriptions are identical. (Public key validation was voluntarily omitted in the HMQV design, but in this case, the protocol is already known to be insecure [13, 14]).

In Attack 4, as $XA^d = G^{td}$, $(XA^d)^{y+eb} = (YB^e)^{td}$ the attacker derives the same secret as $\hat{B}$. Even if [22] considers the attack with $t = 0$, for this value of t the attack cannot work in effect. In MQV the shared secret is tested to be non–identity; and in the HMQV standards (drafts), the shared secret is also tested to be non–identity (see [8, pp 43–46], for instance). With other small values of t $(1, 2, 3, \cdots)$ the attack can be mounted.

Attack 4 EDA Attack against MQV

```
(1) Choose $X \in \mathcal{G}$ and compute $d = 2^l + X \mod 2^l$.
(2) Compute $A = X^{-d^{-1}} G^t$ for some $t \in \{1, 2, \cdots\}$, and register the static key
    $A$ on behalf of $\hat{A}$.
(3) Send $(\hat{A}, X)$ to $\hat{B}$.
(4) Intercept $\hat{B}$'s message $(\hat{B}, Y)$.
(5) Compute the secret shared with $\hat{B}$, $\sigma = (YB^e)^{td}$, and derive the same session
    key as $\hat{B}$.
```

An attempt to formalize computational fairness is given in [21, section 6.1]. But, as the attacker can register a static public key of its choice, it is difficult see how computational fairness can be achieved. Indeed, a protocol is said to achieve computational fairness "if for any successfully finished session run by a malicious player (e.g., $\hat{B}$) with an honest player (e.g., $\hat{A}$), the session–key computation involves the same number of (strongly or general) non–malleably independent dominant–operation values for both the honest player and the malicious player." [22, Definition 6.3]. And, [22, Proposition 6.1] claims that the (s)YZ protocol is session key computationally fair under the random oracle model, while (H)MQV is not.

In fact, the formalization of the computational fairness notion is unclear, and it is difficult to see how two–pass protocols can achieve this attribute, as the attacker is allowed to register a static public key of its choice. Consider, for instance, the sYZ protocol; when an attacker registers $A = G$ as static public key, and uses $X = G$ as ephemeral public key, the session key derivation, is "unfair" in the sense that it requires much less operations for the attacker than for an honest party (which chooses its public keys at random). Consider a malicious party which performs as in Attack 5.

Attack 5 Computational Unfairness Attack against the sYZ Protocol

```
(1) Register the static public key $A = G$ to obtain a certificate $\hat{A}$.
(2) Choose $X = G$ as ephemeral public key.
(3) Send $(\hat{A}, A, X)$ to $\hat{B}$
(4) Intercept $\hat{B}$'s response $(\hat{B}, B, Y = g^y)$.
(5) Compute $e = \tilde{H}(\hat{A}, A, \hat{B}, B, X, Y)$, the secret $\sigma = BY^{1+e}$, and the session
    key $K = H(\sigma)$.
```

As $x = a = 1$, it is clear that the attacker computes the same secret as $\hat{B}$. Moreover, the computational effort it requires to execute Attack 5 against the sYZ protocol, one exponentiation, is much less than what is required to execute the EDA attack against MQV (two exponentiations at least, when the simultaneous exponentiation technique [16, section 14.6.1] is used). Naturally, one can ask how does the (s)YZ protocols achieve computational fairness. Again, as the attacker can register the static public key of its choice, and use ephemeral keys of its choice (the keys can be used with small exponents for instance), it is difficult to be how the computational fairness security attribute can be achieved in two pass key agreement protocols.

5 The Deniable Internet Key Exchange (DIKE)

In this section, we recall the Deniable Internet Key Exchange (DIKE) protocol [23], and show that its design is not careful enough. Namely, we show that if an attacker learns the ephemeral private key used by a session initiator, it can impersonate any party (alive or not) to the initiator.

Deniability is a security goal of the DIKE protocol; it is also another goal that if an attacker completes a session, which matching session exists, then it knows not only the ephemeral private key corresponding to its outgoing ephemeral public key, but also the static private key corresponding to its alleged outgoing static public key [23, pp. 331, 337]. Notice that it is widely admitted that *an attacker should not be able to impersonate a party, unless it knows the party's static private key*; and the use of ephemeral private keys in session derivation processes is also common, in particular to achieve (weak) forward secrecy, or to avoid key replication attacks.

An execution of DIKE, with initiator $\hat{A}$ and responder $\hat{B}$, is as in Protocol 6, if any verification fails, the execution aborts.

Protocol 6 The DIKE Protocol in the main model

```
I) At session activation with identifier $sid$, $\hat{A}$ does the following:
   (a) Choose $x \in_R [1, q-1]$ and compute $X = G^x$.
   (b) Send $(sid, X)$ to the peer.
II) At receipt of $(sid, X)$, the responder $\hat{B}$ does the following:
   (a) Verify that $X \in \mathcal{G}^*$.
   (b) Choose $y \in_R [1, q-1]$ and compute $Y = G^y$.
   (c) Compute $t_B = H(sid, \hat{B}, Y, X, X^y)$.
   (d) Send $(sid, \hat{B}, Y, t_B)$ to the initiator.
III) At receipt of $(sid, \hat{B}, Y, t_B)$, $\hat{A}$ does the following:
   (a) Verify that $Y \in \mathcal{G}^*$.
   (b) Verify that $t_B = H(sid, \hat{B}, Y, X, Y^x)$.
   (c) Compute $\tau_A = H(sid, \hat{A}, X, Y, Y^a, Y^x)$.
   (d) Send $(sid, \hat{A}, \tau_A)$ to $\hat{B}$.
IV) At receipt of $(sid, \hat{A}, \tau_A)$, $\hat{B}$ does the following:
   (a) Verify that $\tau_A = H(sid, \hat{A}, X, Y, A^y, X^y)$.
   (b) Compute $\tau_B = H(sid, \hat{B}, Y, X, X^b, X^y)$.
   (c) Send $(sid, \tau_B)$ to $\hat{A}$.
   (d) Compute $K = H_K(X^y, X, Y)$.
V) At receipt of $(sid, \tau_B)$, $\hat{A}$ does the following:
   (a) Verify that $\tau_B = H(sid, \hat{B}, Y, X, B^x, Y^x)$.
   (b) Compute $K = H_K(X^y, X, Y)$.
VI) The shared session key is $K$.
```

For the security arguments of the DIKE protocol, Yao and Zhao introduce a new assumption termed the CKEA assumption. They also introduce a simulation based security definition, which aims to model non–malleability for Diffie–Hellman key exchange protocols in concurrent settings. The DIKE protocol is shown CK secure and tag–based robust non–malleable in the restricted RO model, under the Gap Diffie–Hellman and the CKEA assumptions.

Failure in Authentication

Despite its security arguments, and the use of the non standard CKEA assumption, the DIKE protocol fails in authentication, which is a primary goal of key exchange. "A minimal requirement for a secure key–exchange protocol is that the attacker, not knowing the private key of a party $\hat{A}$, should not be able to impersonate $\hat{A}$" [10, p. 14]. This requirement is also considered as minimal in [7, section 2]. In the DIKE protocol, intended "to add novelty and new value to the IKE key exchange standard" [23, p. 320], if an attacker learns the ephemeral private key in a session initiated at $\hat{A}$, it can impersonate any party to $\hat{A}$. The impersonation is described in Attack 7. Notice that ephemeral private key leakage is a realistic assumption, as in many applications the pairs (x, G^x) are off–line precomputed and kept in less protected storage than the long–lived private keys.

Attack 7 Impersonation Attack against DIKE

```
When the initiator, $\hat{A}$ is activated with a session identifier $sid$, the attacker does
the following:
(1) Intercept $\hat{A}$'s message to the responder $(sid, X)$.
(2) Learn the ephemeral private key in the session at $\hat{A}$.
(3) Choose $y \in_R [1, q-1]$ and compute $Y = G^y$.
(4) Compute $t_B = H(sid, \hat{B}, Y, X, X^y)$.
(5) Send $(sid, \hat{B}, Y, t_B)$ to the initiator.
(6) Intercept $\hat{A}$'s message to $\hat{B}$ $(sid, \hat{A}, \tau_A)$.
(7) Verify that $\tau_A = H(sid, \hat{A}, X, Y, A^y, X^y)$.
(8) Compute $\tau_B = H(sid, \hat{B}, Y, X, B^x, Y^x)$.
(9) Send $(sid, \tau_B)$ to $\hat{A}$.
(10) Compute and use $K = H_K(X^y, X, Y)$ to communicate with $\hat{A}$ on be-
     half of $\hat{B}$.
```

The session at $\hat{A}$ does not abort, as both t_B and τ_B are valid tags, so $\hat{A}$'s verifications at steps IIIb and Va of Protocol 6 do not fail. It is also clear that (*i*) the attacker derives the same session key as $\hat{A}$, (*ii*) the attacker has no knowledge of $\hat{B}$'s static private key, and (*iii*) $\hat{A}$ believes its session key is shared with $\hat{B}$. This is clearly a *failure in authentication.* This attack illustrates not only the insufficiencies in the DIKE design, but also the impossibility of capturing some kind of impersonation attacks in the CK–model. Indeed, in the CK model, in contrary to the (s)eCK models for instance, once the ephemeral information in a session is exposed, the model does not care about the security of the session, even if the matching session or intended peer does not exist. As a consequence, the CK model cannot guarantee that an attacker cannot impersonate a party unless it knows its static private key.

Notice also that such an attack cannot hold against the SIGMA protocol, as in an execution of SIGMA, a peer has to sign a fresh nonce; this cannot be done without a knowledge of the static private corresponding to its alleged public key. Instead of adding novelty and new value to the IKE standard, [22] proposes a weaker protocol.

The SMQV–1 protocol

We propose here a protocol inspired from the SMQV protocol [18], to reaffirm the usefulness of the building blocks of the SMQV protocol. A similar variant can be obtained from the FHMQV protocol.

The protocol's design follows the same principles as that of the SMQV protocol. The shared secret is a dual signature, from which the session key is computed. And, in accordance with the IKEv2 design choices [7], we make the initiator reveal its identity first. Notice also that it is straightforward to modify the protocol to make the responder reveal its identity first. We do not use the CK session identifiers and matching sessions definitions (owing to their limitations); instead, we use the session identifiers from the combined CK model [15]. Sessions at $\hat{A}$ are activated with parameters $(\hat{A}, \tilde{B})$ or $(\tilde{A}, \hat{B}, in)$ wherein $\tilde{A}$ and $\tilde{B}$ are *destination addresses*, and in is an incoming message.

Protocol 8 The SMQV–1 Protocol

I) At session activation with parameter $(\tilde{A}, \tilde{B})$, $\hat{A}$ does the following:
   (a) Choose $x \in_R [1, q-1]$ and compute $X = G^x$.
   (b) Send $(\tilde{B}, \hat{A}, X)$ to $\tilde{B}$.
II) At receipt of $(\tilde{B}, \hat{A}, X)$, the responder $\hat{B}$ does the following:
   (a) Verify that $X \in \mathcal{G}^*$.
   (b) Choose $y \in_R [1, q-1]$ and compute $Y = G^y$.
   (c) Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B})$ and $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.
   (d) Compute $s_B = ye + b$ and $\sigma = (X^d A)^{s_B}$.
   (e) Compute $\tau_B = H(Y, X, \sigma)$.
   (f) Send $(\hat{A}, \hat{B}, Y, \tau_B)$ to $\hat{A}$.
III) At receipt of $(\hat{A}, \hat{B}, Y, \tau_B)$, $\hat{A}$ does the following:
   (a) Verify that $Y \in \mathcal{G}^*$.
   (b) Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B})$ and $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.
   (c) Compute $s_A = xe + a$ and $\sigma = (Y^b B)^{s_A}$.
   (d) Verify that $\tau_B = H(Y, X, \sigma)$.
   (e) Compute $\tau_A = H(X, \sigma, Y)$.
   (f) Send $(sid, \hat{A}, \tau_A)$ to $\hat{B}$.
   (g) Compute $K = H(\sigma, X, Y)$.
IV) At receipt of $(sid, \hat{A}, \tau_A)$, $\hat{B}$ does the following:
   (a) Verify that $\tau_A = H(X, \sigma, Y)$.
   (b) Compute $K = H(\sigma, X, Y)$.
V) The shared session key is K.

In a SMQV–1 execution, both parties confirm their ability to compute the secret group element σ. The inputs order in the tags computation is set (Y, X, σ) for $\hat{A}$

and (X, σ, Y) for $\hat{B}$ to avoid reflection attacks. The SMQV–1 protocol is more efficient than the DIKE protocol; when ephemeral keys are off–line pre–computed, and execution of SMQV–1 requires 1.25 times a single exponentiation, when the simultaneous exponentiation technique [16, section 14.6.1] is used. Also, contrary to DIKE, the protocol is not vulnerable to the impersonation attack we propose.

The SMQV–1 protocol is secure in the combined eCK model and is deniable, under the gap Diffie–Hellman assumption and the Random Oracle model. For lack of space, we do not provide here a full security analysis of the protocol, this will be given in the full version of this paper. Instead, in the appendix, we give a full–fledged proof of a variant of the FXCR–1 scheme, and show that the critics on SMQV from [24] missed some important points.

6 Conclusion

We discussed some of the weaknesses of the CK and CK_{HMQV} security models. We showed that the (s)YZ protocols do not achieve their claimed CK_{HMQV} security. We also showed that the (s)YZ protocols do not achieve the computational fairness attribute; our attack suggests also that no two–pass protocol can achieve this attribute. We showed that the Deniable Internet Key Exchange (DIKE) fails in authentication. Besides this failure in authentication, our attack illustrates again the inability of capturing some kind impersonation attacks in the CK model. We proposed a variant of the SMQV protocol, the SMQV–1, geared to the post peer specified model. The SMQV protocol is secure in the combined eCK model and deniable.

In a forthcoming stage, we will be interested in clarifying and illustrating the weaknesses of the (e)CK and CK_{HMQV} models. We will also work on an adaptation of the seCK model to the post peer specified model, and on illustrating the differences between the (e)CK, CK_{HMQV}, and seCK securty models.

References

[1] Bellare M., Rogaway P.: Entity Authentication and Key Distribution. In Proc. of Crypto 93, Lecture Notes in Computer Science, vol. 773, pp. 232–249, Springer–Verlag, 1993.

[2] Boyd C., Mathuria A.: Protocols for Authentication and Key Establishment. Springer Verlag, 2003.

[3] Canetti R., Krawczyk H.: Analysis of Key–Exchange Protocols and Their Use for Building Secure Channels. In Proc. of Eurocrypt 01, Lecture Notes in Computer Science, vol. 2045, pp. 453–474, Springer–Verlag, 2001.

[4] Choo K.–K. R.: Refuting the Security Claims of Mathuria and Jain (2005) Key Agreement Protocols International Journal of Network Security, Vol.7, No.1, pp.15–23, July 2008.

[5] Cremers C.: Formally and Practically Relating the CK, CK–HMQV, and eCK Security Models for Authenticated Key Exchange. Cryptology ePrint Archive, 2009/253, 2009.

[6] Cremers C.: Examining Indistinguishability–Based Security Models for Key Exchange Protocols: The case of CK, CK-HMQV, and eCK. In Proc. of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, 2011.

[7] Harkins D., Kaufman C., Kivinen T., Kent S., Perlman R.: Design Rationale for IKEv2. IPSec Working Group Internet Draft, available at `http://tools.ietf.org/html/draft-ietf-ipsec-ikev2-rationale-00`, 2002.

[8] IEEE P1363: Draft Standard for Public Key Cryptography. IEEE, 2009.

[9] Krawczyk H.: SIGMA: The 'SiGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In Proc. of Crypto 03, Lecture Notes in Computer Science, vol. 2729, pp. 400–425, Springer–Verlag, 2003.

[10] Krawczyk H.: HMQV: A Hight Performance Secure Diffie–Hellman Protocol. In Proc. of Crypto 05, Lecture Notes in Computer Science, vol. 3621, pp. 546–566, Springer–Verlag, 2005.

[11] LaMacchia B., Lauter K., Mityagin A.: Stronger Security of Authenticated Key Exchange. In Proc of ProvSec 2007, Lecture Notes in Computer Science, vol. 4784, pp. 1–16, Springer–Verlag, 2007.

[12] Maurer U. M., Wolf S.: Diffie–Hellman Oracles. In Proc. of Crypto 96, Lecture Notes in Computer Science, vol. 1109, pp. 268–282, Springer–Verlag, 1996.

[13] Menezes A., Ustaoglu B.: On the Importance of Public–Key Validation in the MQV and HMQV Key Agreement Protocols. In Proc. of Indocrypt 2006, Lecture Notes in Computer Science, vol. 4329, pp. 133-147, Springer–Verlag, 2006.

[14] Menezes A.: Another Look at HMQV. Journal of Mathematical Cryptology, vol. 1, pp. 148–175, Walter de Gruyter, 2007.

[15] Menezes A., Ustaoglu B.: Comparing the Pre– and Post–specified Peer Models for Key Agreement. International Journal of Applied Cryptography, vol. 1(3) pp. 236–250, 2009.

[16] Menezes A., van Oorschot P., Vanstone S.: Handbook of Applied Cryptography. CRC Press, 1996.

[17] Pointcheval D., Stern J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology, vol. 13, pp. 361–396, Springer–Verlag, 2000.

[18] Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A New Security Model for Authenticated Key Agreement. In Proc. of the 7th International Conference on Security and Cryptography for Networks — SCN 2010, Lecture Notes in Computer Science, vol. 6280, pp. 219–234, Springer–Verlag, 2010.

[19] Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A New Security Model for Authenticated Key Agreement. Cryptology ePrint Archive: Report 2010/237.

[20] Yao, A.C., Zhao, Y.: Method and Structure for Self–Sealed Joint Proof–of–Knowledge and Diffie-Hellman Key-Exchange Protocols. PCT 2009, available at `http://www.wipo.int/patentscope/search/en/detail.jsf;jsessionid=C14F61855C476745B13CFDB74D848875.wapp2?docId=WO2009056048&recNum=1&tab=PCTDocuments&maxRec=&office=&prevFilter=&sortOption=&queryString=`. Accessed on 26 September 2011.

[21] Yao, A.C., Zhao, Y.: A New Family of Practical Non-Malleable Protocols. Cryptology ePrint Archive: Report 2011/035.

[22] Yao, A.C., Zhao, Y.: A New Family of Practical Non-Malleable Protocols. CoRR abs/1105.1071, 2011.

[23] Yao, A.C., Zhao, Y.: Deniable Internet Key Exchange. In Proc. of Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Lecture Notes in Computer Science, vol. 6123, pp. 329–348, 2010.

[24] Yoneyama K. and Zhao Y.: Taxonomical Security Consideration of Authenticated Key Exchange Resilient to Intermediate Computation Leakage. Lecture Notes in Computer Science, vol. 6980, pp. 348–365, Springer–Verlag, 2011.

A On the Critics of the SMQV Arguments from [24]

In [24], Yoneyama and Zhao propose a taxonomy of authenticated key agreement protocols in regard with their security arguments and assumptions. We have already showed that this taxonomy is partly flawed, The srYZ is classified as seCK–secure, while this is erroneous. Again, their claim that SMQV and FHMQV are not secure even in the CK_{HMQV} model, and then are insecure in the seCK, suggests that seCK security should imply CK or CK_{HMQV} security. This is a misunderstanding. In [18, 19], the authors claimed that "the seCK model is *practically* stronger than the CK model" and that the "seCK model encompasses the eCK one", meaning that the seCK–model captures more security attributes than the CK–model, and that seCK security formally implies eCK–security.

The security of the SMQV and FHMQV protocols depend on that of the FXCR–1 scheme. Yoneyama and Zhao [24] consider the security of the FXCR–1 scheme in the case the signer is the initiator. Strictly speaking, this defines another scheme, as in the definition of the FXCR–1 scheme, the communications are initiated by the verifier (not the signer); and the signature (Y, X^{s_B}) is provided to the verifier *after* it provides its challenge X. When the interaction order is changed, this defines another scheme (we call FXCR–2), and the proof may become invalid. Here, we show that even in this modified variant of the FXCR–1 scheme, the security remains valid.

Definition 3 (FXCR–2 Signature)**.** Let $\hat{B}$ be a party with public key $B \in \mathcal{G}^*$, and $\hat{A}$ a verifier; $\hat{B}$'s signature on a message m and challenge X, provided by $\hat{A}$ ($x \in_R [1, q-1]$ is chosen and kept secret by $\hat{A}$), *after it receives the ephemeral element* Y, is $Sig_{\hat{B}}(m, X) = (Y, X^{s_B})$, where $Y = G^y$, $y \in_R [1, q-1]$ is chosen by $\hat{B}$, and $s_B = ye + b$, wherein $e = \bar{H}(Y, X, m)$. And, $\hat{A}$ accepts the pair (Y, σ_B) as a valid signature if $Y \in \mathcal{G}^*$ and $(Y^e B)^x = \sigma_B$.

The security of the FXCR–2 scheme is given by the following proposition.

Proposition 1 (FXCR–2 Security)**.** *Under the CDH assumption in $\mathcal{G}$ and the RO model, there is no adaptive probabilistic polynomial time attacker, which given a public key B, a challenge X_0 ($B, X_0 \in_R \mathcal{G}^*$), together with a hashing and a FCXR–2 signing oracles, outputs with non–negligible success probability a triple (m_0, Y_0, σ_0) such that:*

(1) (Y_0, σ_0) *is a valid signature with respect to the public key B, and the message–challenge pair (m_0, X_0); and*

(2) (Y_0, σ_0) *was not obtained from the signing oracle with a query on* (m_0, X_0)*.*

Proof. There are two classes of attackers we distinguish in our analysis. Suppose an attacker, which does the following at some point of its execution:
(1) activate $\hat{B}$ with a message m to obtain Y,
(2) issue digest queries on (Y, Z_i, m), for arbitrary Z_is,
(3) send Z_{i_0} to $\hat{B}$, where Z_{i_0} equals some Z_i,
(4) receive the signature (Y, σ, s_B).
Notice that the attacker is given s_B in addition to the signature σ. In this sequence, as the digest value e has to be set before the incoming ephemeral public key is known, we cannot simulate consistently the disclosure of s_B to the verifier. We summarize the sequence of query in Seq1 below. Without loss of generality, we omit the possible independent computations the attacker may perform between two consecutive steps of Seq1.

Algorithm 9 Seq1

```
(1) Activate the signer $\hat{B}$ with a message $m$ to obtain $Y$.
(2) Issue digest queries on $(Y, Z_i, m)$, for an arbitrary $Z_i \in \mathcal{G}^*$.
(3) Send $Z_{i_0}$ to $\hat{B}$, where $Z_{i_0}$ equals some $Z_i$.
(4) Receive the signature on message $m$ and challenge $Z_{i_0}$ $(Y, \sigma, s_B)$.
```

Let $\mathfrak{B}$ be the family of polynomial time attackers which at some point of their run, execute Seq1 (the attackers may execute Seq1 many times).

Let $\mathcal{A} \notin \mathfrak{B}$ be a polynomial time attacker which, given $B, X_0 \in_R \mathcal{G}^*$, succeeds with non–negligible probability in forging a fresh and valid signature, with respect to the public key B and challenge X_0. Let Q_h and Q_s be respectively the number of queries that $\mathcal{A}$ asks to the hashing and signing oracles. Using $\mathcal{A}$ we build a polynomial time CDH solver $\mathcal{S}$ which succeeds with non–negligible probability. The solver $\mathcal{S}$ provides $\mathcal{A}$ with random coins, and simulates the digest and signature queries. The interactions between $\mathcal{S}$ and $\mathcal{A}$ are detailed in Figure 10.

Under the RO model, the distribution of the simulated signatures is indistinguishable from that of real signatures generated by $\hat{B}$, except the deviation that occurs when the same Y is chosen twice. Since the number of queries to the oracles is less than $(Q_h + Q_s)$, and Y is chosen uniformly at random in $\mathcal{G}$, this deviation occurs with probability less than $(Q_h + Q_s)/q$, which is negligible. Hence this simulation is perfect, except with negligible probability. Moreover the probability of producing a valid forgery without querying $\bar{H}(Y_0, X_0, m_0)$ is 2^{-l}. Thus under this simulation, $\mathcal{A}$ outputs with non–negligible probability a valid forgery $(Y_0, X_0, m_0, \sigma_0)$; we denote $\bar{H}(Y_0, X_0, m_0)$ by e_0.

From the Forking Lemma [17], the repeat experiment outputs with non–negligible probability a valid forgery $(Y_0, X_0, m_0, \sigma_0')$ with a digest e_0', which with probability $1 - 2^{-l}$ is different from e_0. Then the computation

$$\left({\sigma_0}^{{e_0}^{-1}} / {\sigma_0'}^{{e_0'}^{-1}}\right)^{\left({e_0}^{-1} - {e_0'}^{-1}\right)^{-1}} = \left(\frac{\left(Y_0 B^{{e_0}^{-1}}\right)^{x_0}}{\left(Y_0 B^{{e_0'}^{-1}}\right)^{x_0}}\right)^{\left({e_0}^{-1} - {e_0'}^{-1}\right)^{-1}} = B^{x_0}$$

Figure 10 CDH solver from $\mathcal{A}$

Run of $\mathcal{A}$:

(a) At $\mathcal{A}$'s digest query on (Y, X, m), $\mathcal{S}$ responds as follows: *(i)* if a value is already assigned to $\bar{H}(Y, X, m)$, $\mathcal{S}$ returns $\bar{H}(Y, X, m)$; *(ii)* otherwise $\mathcal{S}$ responds with $e \in_R \{0,1\}^l$, and sets $\bar{H}(Y, X, m) = e$.

(b) When $\mathcal{S}$ is activated with a message m, it does the following: (i) Choose $s_B \in_R [1, q-1]$, $e \in_R \{0,1\}^l$, set $Y = (G^{s_B} B)^{e^{-1}}$ and $\bar{H}(Y, \star, m) = e$. (The inputs yielding to e are set temporarily to $(Y, \star, m)$ and updated once X is known.) If Y was previously chosen as ephemeral key, $\mathcal{S}$ aborts. (ii) Responds with (Y, m).

(c) At $\mathcal{A}$'s signature query on (Y, m, X), $\mathcal{S}$ responds as follows: *(i)* Update $\bar{H}(Y, \star, m) = e$ to $\bar{H}(Y, X, m) = e$. *(ii)* Responds with (Y, X^{s_B}, s_B) (s_B is given in addition to X^{s_B}).

(d) At $\mathcal{A}$'s halt, $\mathcal{S}$ verifies that $\mathcal{A}$'s output $(Y_0, X_0, m_0, \sigma_0)$ (if any) satisfies the following conditions. If one of these conditions is not satisfied $\mathcal{S}$ aborts.
- $Y_0 \in \mathcal{G}^*$ and $\bar{H}(Y_0, X_0, m_0)$ was queried from $\bar{H}$.
- The signature (Y_0, σ_0) was not returned by $\hat{B}$ on query (m_0, X_0).

Repeat: $\mathcal{S}$ executes a new run of $\mathcal{A}$, using the same input and coins; and answering to all digest queries before $\bar{H}(Y_0, X_0, m_0)$ with the same values as in the previous run. The new query of $\bar{H}(Y_0, X_0, m_0)$ and subsequent queries to $\bar{H}$ are answered with new random values.

Output: If $\mathcal{A}$ outputs a second signature on $(Y_0, X_0, m_0, \sigma_0')$ satisfying conditions of step d, with a hash value $\bar{H}(Y_0, X_0, m_0)_2 = e_0' \neq e_0 = \bar{H}(Y_0, X_0, m_0)_1$, then $\mathcal{S}$ outputs

$$\left({\sigma_0}^{{e_0}^{-1}} / {\sigma_0'}^{{e_0'}^{-1}}\right)^{({e_0}^{-1} - {e_0'}^{-1})^{-1}}$$

as a guess for $CDH(B, X_0)$.

gives $CDH(B, X_0)$. Recall that such a polynomial CDH solver, succeeding with non–negligible probability, can be transformed into an efficient CDH solver [12].

For attackers in $\mathfrak{B}$, we do not provide a direct simulation; instead, we show that their success probability is bounded by that of a class of attackers which can be efficiently simulated.

Let $\mathcal{B}$ be an attacker in $\mathfrak{B}$, and $\mathfrak{d}(|q|)$ and $\mathfrak{m}(|q|)$ be respectively upper bounds on the number of Z_i the attacker chooses at step 2 of Seq1, and the number of times $\mathcal{B}$ executes Seq1. For simplicity (in the notations), we suppose that whenever $\mathcal{B}$ executes Seq1, it chooses $\mathfrak{d}(|q|)$ Z_is at step 2.

For all $\mathcal{B} \in \mathfrak{B}$, let $\mathcal{B}_R$ be an attacker, which receives in addition to $\mathcal{B}$'s input, the resource vector $\mathbf{v} = \Big((i_1^0, \cdots, i_{\mathfrak{m}}^0), (Z_{11}, \cdots, Z_{1\mathfrak{d}}), \cdots, (Z_{\mathfrak{m}1}, \cdots, Z_{\mathfrak{m}\mathfrak{d}})\Big)$, where $Z_{ij} \in_R \mathcal{G}^*$ and $i_i^0 \in_R [1, \mathfrak{d}]$, and performs *exactly* the same way as $\mathcal{B}$, except that whenever $\mathcal{B}$ executes the sequence of queries Seq1 for the l–th time, $\mathcal{B}_R$ executes the modified sequence Seq2, using i_l^0 and $(Z_{l1}, \cdots, Z_{l\mathfrak{d}})$. And, when $\mathcal{B}$ uses, for any other computation Z_i, chosen during the l–th execution Seq1, $\mathcal{B}_R$ uses Z_{li}.

Algorithm 11 Seq2

```
(1) Activate the signer $\hat{B}$ with a message $m$ to obtain $Y$ ($\mathcal{B}_R$ uses the same
    message as $\mathcal{B}$).
(2) Issue digest queries on $(Y, Z_{li}, m)$, for $Z_{li} \in \{Z_{l1}, \cdots, Z_{l\mathfrak{d}}\}$.
(3) Send $Z_{i_l^0}$ to $\hat{B}$.
(4) Receive the signature on message $m$ and challenge $Z_{i_l^0}$ $(Y, \sigma, s_B)$.
```

Notice that if $\mathcal{B}$ is polynomial, then so is $\mathcal{B}_R$. Let $\mathbf{V}$ be the set of resource vectors. For $\mathbf{v} \in \mathbf{V}$, we say that $\mathcal{B}_R(\mathbf{v})$ *matches* $\mathcal{B}$, if for all $l \in [1, \mathfrak{m}]$, the l–th time $\mathcal{B}$ executes Seq1, it chooses $\{Z_{l1}, \cdots, Z_{l\mathfrak{d}}\}$ at step 2, and sends Zi_l^0 at step 3. It is clear that if $\mathcal{B}_R(\mathbf{v})$ matches $\mathcal{B}$, then the success probability of $\mathcal{B}$ is,

$$\Pr(\mathsf{Succ}_{\mathcal{B}}) = \Pr(\mathsf{Succ}_{\mathcal{B}_R(\mathbf{v})}).$$

For $\mathcal{B} \in \mathfrak{B}$, we say $\mathbf{v} \in \mathbf{V}$ *possible* if there is non–zero probability that $\mathcal{B}_R(\mathbf{v})$ matches $\mathcal{B}$. Let $Poss(\mathbf{V})$ denote the set of possible resource vectors. For every run of $\mathcal{B}$, there is some $\mathbf{v} \in Poss(\mathbf{V})$ such that $\mathcal{B}_R(\mathbf{v})$ matches $\mathcal{B}$ ($\mathbf{v}$ can be built from the choices of $\mathcal{B}$ in its executions of Seq1), hence

$$\Pr(\mathsf{Succ}_{\mathcal{B}}) \leqslant \max_{\mathbf{v} \in Poss(\mathbf{V})} \Pr(\mathsf{Succ}_{\mathcal{B}_R(\mathbf{v})}).$$

Now, it suffices to show that for all $\mathcal{B}$ and all $\mathbf{v} \in Poss(\mathbf{V})$ $\Pr(\mathsf{Succ}_{\mathcal{B}_R(\mathbf{v})})$ is negligible. For this purpose, we modify the simulation to take as input $\mathbf{v}$ and respond as follows, when $\mathcal{B}_R(\mathbf{v})$ executes the sequence Seq2 for the l–th time.

- When $\mathcal{B}_R$ activates the signer $\hat{B}$ with a message m to obtain Y, the simulator $\mathcal{S}$ does the following:
  - Choose $s_B \in_R [1, q-1], e \in_R \{0,1\}^l$, set $Y = (G^{s_B} B^{-1})^{e^{-1}}$ and $e = H_1(Y, Z_{i_l^0}, m)$.
  - Respond with (Y, m).
- At $\mathcal{B}_R$'s signature query on $(Y, Z_{i_l^0}, m,)$, $\mathcal{S}$ responds with $(Z_{i_l^0}, {Z_{i_l^0}}^{s_B}, s_B)$.

The simulation is polynomial–time and consistent for all $\mathbf{v} \in Poss(\mathbf{V})$ and $\mathcal{B}_R$. As $\mathcal{S}$ knows, from the resource vector, what will be the incoming challenge, answers to the queries of Seq2 are consistently simulated. If for some $\mathbf{v}$, $\Pr(\mathsf{Succ}_{\mathcal{B}_R(\mathbf{v})})$ is non–negligible, using the oracle replay technique, we build a CDH solver which succeeds with non–negligible probability. Hence under the CDH assumption, $\Pr(\mathsf{Succ}_{\mathcal{B}_R(\mathbf{v})})$ is negligible, for all $\mathbf{v} \in Poss(\mathbf{V})$. This implies $\Pr(\mathsf{Succ}_{\mathcal{B}})$ is negligible. □