A New Security Model for Authenticated Key Agreement

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

[1]Netheos R&D <aug.sarr@gmail.com>
[1,2]Institut Fourier – CNRS, Université Grenoble 1 <Philippe.Elbaz-Vincent@ujf-grenoble.fr>
[3]LIP6 – CNRS, Université Paris 6 <Jean-Claude.Bajard@lip6.fr>

Abstract

The Canetti–Krawczyk (CK) and extended Canetti–Krawczyk (eCK) security models, are widely used to provide security arguments for key agreement protocols. We discuss security shades in the (e)CK models, and some practical attacks unconsidered in (e)CK–security arguments. We propose a strong security model which encompasses the eCK one. We also propose a new protocol, called Strengthened MQV (SMQV), which in addition to provide the same efficiency as the (H)MQV protocols, is particularly suited for distributed implementations wherein a tamper–proof device is used to store long–lived keys, while session keys are used on an untrusted host machine. The SMQV protocol meets our security definition under the Gap Diffie–Hellman assumption and the Random Oracle model.

Keywords: authenticated key agreement, practical vulnerability, strengthened eCK model, SMQV.

1 Introduction

Much of recent research on key agreement deals with provably secure key exchange. Since this approach was pioneered by Bellare and Rogaway [1], different models were proposed [3, 5, 31, 7, 14, 17]. Among these models, the Canetti–Krawczyk (CK) [7] and extended Canetti–Krawczyk (eCK) [17] models (which are incomparable [10, 34]) are considered as “advanced” approaches to capture security of key agreement protocols; and security arguments for recent protocols are usually provided in the (e)CK models.

Broadly, a security model specifies, among other things, what constitutes a security failure, and what adversarial behaviors are being protected against. The aim is that a protocol shown secure, in the model, confines to the minimum the effects of the considered adversarial behaviors. In the CK and eCK models, session specific information leakages are respectively captured using reveal queries on *session states* and *ephemeral keys*, which stores session specific information; the adversary is supposed to interact with parties, and to try to distinguish a session key from a randomly chosen value. A protocol is secure if an adversary controlling communications between parties, cannot distinguish a session key from a random value, unless it makes queries which overtly reveal the session key.

Unfortunately, adversaries do not always behave as expected. When leakages on intermediate results in computing session keys are considered, (e)CK–secure protocols often fail in authentication; and the widely accepted principle that *an attacker should not be able to impersonate a party, unless it knows the party’s static key* is not achieved. This makes clearly desirable a security model, which captures intermediate results leakages resilience, in addition to the security attributes considered in the (e)CK models.

From [30], we have a hybrid security definition, which considers leakages on intermediate results; however the model cannot be shown to encompass the CK or eCK models. In addition, the security definition from [30] considers intermediate results and ephemeral key leakages in separate settings. In this paper, we propose a strong security definition, the *strengthened* eCK (seCK) model, which encompasses the eCK model, and considers leakages on intermediate results in computing session keys. We also propose a new protocol called Strengthened MQV (SMQV). The SMQV protocol provides the same efficiency as the (H)MQV protocols [18, 14]. In addition, because of its resilience to intermediate results leakages, SMQV is particularly suited for implementations using a tamper–resistant device, to store the static keys, together with a host machine on which sessions keys are used. In such SMQV implementations, the non–idle time computational effort of the device can be securely reduced to few non–costly operations.

This paper is organized as follows. In section 2 we discuss security shades in the (e)CK models. The protocol $\mathcal{P}$ [24] is described as an example of protocol that is formally CK–secure, but practically insecure, unless session identifiers are added with further restrictions. We also discuss the vulnerability of the NAXOS type protocols to ephemeral Diffie–Hellman (DH) exponent leakages. In section 3 we present the *strengthened* eCK (seCK) model. In section 4, we describe the SMQV protocol, which meets the seCK security definition, and its building blocks. We conclude in section 5.

The following notations are used in this paper: $\mathcal{G}$ 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). If $\hat{A} \neq \hat{B}$, we suppose that no substring of $\hat{A}$ equals $\hat{B}$. H is a 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." The Computational Diffie–Hellman (CDH) assumption is supposed to hold in $\mathcal{G}$; namely, given $U = G^u$ and $V = G^v$ with $U, V \in_R \mathcal{G}^*$, computing $CDH(U, V) = G^{uv}$ is infeasible.

2 Practical Limitations in the (e)CK Models

In this section, we discuss security shades in the (e)CK models, and the related unconsidered attacks. (Please, refer to [10, 34] for outlines and comparisons of the CK and eCK models, or [7, 17] for details.)

Practical Inadequacy of the CK Matching Sessions Definition. In the CK model, two sessions with activation parameters $(\hat{P}_i, \hat{P}_j, s, role)$ and $(\hat{P}_j, \hat{P}_s, s', role')$ are said to be matching if they have the same identifiers ($s = s'$). The requirement about the identifiers (id) used at a party is that *"the session id's of no two KE sessions in which the party participates are identical"* [7]. Session identifiers may, for instance, be nonces generated by session initiators and provided to the peers through the first message in the protocol. In this case, when each party stores the previously used identifiers and verifies at session activation that the session identifier was not used before, the requirement that a party never uses the same identifier twice is achieved.

Unfortunately, when a party, say $\hat{B}$, has no mean to be aware of the sessions initiated at the other parties, and intended to it, apart from receiving the initiator's message, the CK model insufficiently captures impersonations attacks. Consider, for instance, Protocol 1 (wherein H and H_2 are digest functions); it is from [24], and is CK–secure under the Gap Diffie–Hellman assumption [21] and the Random Oracle (RO) model [2]. As the session

Protocol 1 The protocol $\mathcal{P}$

```
I) At session activation with parameters $(\hat{A}, \hat{B}, s)$, $\hat{A}$ does the following:
    (a) Create a session with identifier $(\hat{A}, \hat{B}, s, \mathcal{I})$.
    (b) Choose $x \in_R [1, q-1]$.
    (c) Compute $X = G^x$ and $t_A = H_2(B^a, \mathcal{I}, s, \hat{A}, \hat{B}, X)$.
    (d) Send $(\hat{B}, \hat{A}, s, X, t_A)$ to $\hat{B}$.
II) At receipt of $(\hat{B}, \hat{A}, s, X, t_A)$, $\hat{B}$ does the following:
    (a) Verify that $X \in \mathcal{G}^*$.
    (b) Create a session with identifier $(\hat{B}, \hat{A}, s, \mathcal{R})$.
    (c) Compute $\sigma = A^b$ and verify that $t_A = H_2(\sigma, \mathcal{I}, s, \hat{A}, \hat{B}, X)$.
    (d) Choose $y \in_R [1, q-1]$.
    (e) Compute $Y = G^y$, $t_B = H_2(\sigma, \mathcal{R}, s, \hat{B}, \hat{A}, Y)$, and $K = H(X^y, X, Y)$.
    (f) Destroy $y$ and $\sigma$, and send $(\hat{A}, \hat{B}, s, \mathcal{I}, Y, t_B)$ to $\hat{A}$.
    (g) Complete $(\hat{B}, \hat{A}, s, \mathcal{R})$ by accepting $K$ as the session key.
III) At receipt of $(\hat{A}, \hat{B}, s, \mathcal{I}, Y, t_B)$, $\hat{A}$ does the following:
    (a) Verify the existence of an active session with identifier $(\hat{A}, \hat{B}, s, \mathcal{I})$.
    (b) Verify that $Y \in \mathcal{G}^*$.
    (c) Verify that $t_B = H_2(B^a, \mathcal{R}, s, \hat{B}, \hat{A}, Y)$.
    (d) Compute $K = H(Y^x, X, Y)$.
    (e) Destroy $x$, and complete $(\hat{A}, \hat{B}, s, \mathcal{I})$, by accepting $K$ as the session key.
```

state is defined to be the ephemeral DH exponent[1], while the protocol $\mathcal{P}$ is (formally) CK–secure, its practical security is unsatisfactory, unless session identifiers are added with further restrictions. If session identifiers are nonces generated by initiators, the protocol $\mathcal{P}$ practically fails in authentication. As an illustration, consider Attack 1, wherein the attacker impersonates $\hat{A}$, exploiting a knowledge of an ephemeral DH exponent used at $\hat{A}$.

Attack 1 Impersonation Attack against $\mathcal{P}$ using ephemeral DH exponent leakage

```
I) At the activation of a session $(\hat{A}, \hat{B}, s, \mathcal{I})$, the attacker $\mathcal{A}$ does the following:
    (a) Intercept $\hat{A}$'s message to $\hat{B}$ $(\hat{B}, \hat{A}, s, X, t_A)$.
    (b) Perform a session *SesssionStateReveal* query on $(\hat{A}, \hat{B}, s, \mathcal{I})$ (to obtain $x$).
    (c) Send $(\hat{A}, \hat{B}, s, \mathcal{I}, \bar{1}, 0^{|q|})$ to $\hat{A}$, where $\bar{1}$ is the identity element in $\mathcal{G}$ and $0^{|q|}$ is the
        string consisting of $|q|$ zero bits (as $\bar{1} \not\in \mathcal{G}^*$, $\hat{A}$ aborts the session $(\hat{A}, \hat{B}, s, \mathcal{I})$).
II) When $\mathcal{A}$ decides later to impersonate $\hat{A}$ to $\hat{B}$, it does the following:
    (a) Send $(\hat{B}, \hat{A}, s, X, t_A)$ to $\hat{B}$.
    (b) Intercept $\hat{B}$'s message to $\hat{A}$ $(\hat{A}, \hat{B}, s, \mathcal{I}, Y, t_B)$.
    (c) Compute $K = H(Y^x, X, Y)$.
    (d) Use $K$ to communicate with $\hat{B}$ on behalf of $\hat{A}$.
```

The attacker makes $\hat{B}$ run a session and derive a key with the belief that its peer is $\hat{A}$; in addition, the attacker is able to compute the session key that $\hat{B}$ derives; in practice, this makes the protocol fail in authentication.

The capture of impersonation attacks based on ephemeral DH leakages is insufficient in the CK–model, unless the matching sessions definition is added with further restrictions.

[1] [24] does not specify the information contained in a session state. But, since the adversary controls communications between parties, we do not see another non–superfluous definition of a session state, with which Protocol $\mathcal{P}$ can be shown CK–secure; as the protocol is insecure if the session state is defined to be $\sigma = A^b$.

The reason is that (in a formal analysis) the attacker $\mathcal{A}$ cannot use the session at $\hat{B}$ (in which it impersonates $\hat{A}$) as a test session, since the matching session is exposed, while there is no guarantee that (in practice) $\hat{B}$ would not run and complete such a session. If matching sessions are defined using matching conversations, it becomes clear that Protocol $\mathcal{P}$ is formally and practically insecure. Indeed, in this case, a leakage of an ephemeral DH exponent in a session allows an attacker to impersonate *indefinitely* the session owner to its peer in the exposed session.

On the NAXOS Transformation. In the eCK model [17], all computations performed to derive a session key have to deterministically depend on the ephemeral key, static key, and communication received from the peer.

The design and security arguments of many eCK secure protocols, among which CMQV [33], NAXOS(+, –C) [17, 20, 24], and NETS [19], use the NAXOS transformation [17], which consists in defining the ephemeral DH exponent as the digest of a randomly chosen value and the static private key (of the session owner), and (unnaturally) destroying it after each use. The ephemeral key is then defined to be the random value. However, from a practical perspective, it seems difficult to see how the NAXOS transformation prevents leakages on the ephemeral DH exponents. And, in any environment, which does not guarantee that leakages on DH exponents cannot occur, the NAXOS type protocols security is at best unspecified.

Consider, for instance, Protocol 2, it is from an earlier version[2] of [10]. If the ephemeral keys are defined to be r_A and r_B (as in the NAXOS security arguments [17]) and the signature scheme is secure against chosen message attacks, Protocol 2 can be shown eCK–secure.

Protocol 2 Signed Diffie–Hellman using NAXOS transformation

I) The initiator $\hat{A}$ does the following:
   (a) Choose $r_A \in_R [1, q-1]$, compute $X = G^{H_1(r_A,a)}$, and destroy $H_1(r_A, a)$.
   (b) Compute $\sigma_A = \text{Sig}_{\hat{A}}(\hat{B}, X)$.
   (c) Send $(\hat{B}, X, \sigma_A)$ to $\hat{B}$.

II) $\hat{B}$ does the following:
   (a) Verify that $X \in \mathcal{G}^*$.
   (b) Verify that σ_A is valid with respect to $\hat{A}$'s public key and the message $(\hat{B}, X)$.
   (c) Choose $r_B \in_R [1, q-1]$, compute $Y = G^{H_1(r_B,b)}$, and destroy $H_1(r_B, b)$.
   (d) Compute $\sigma_B = \text{Sig}_{\hat{B}}(Y, \hat{A}, X)$.
   (e) Send $(Y, \hat{A}, X, \sigma_B)$ to $\hat{A}$.
   (f) Compute $K = H_2(X^{H_1(r_B,b)})$.

III) $\hat{A}$ does the following:
   (a) Verify that $Y \in \mathcal{G}^*$.
   (b) Verify that σ_B is valid with respect to $\hat{B}$'s public key and the message $(Y, \hat{A}, X)$.
   (c) Compute $K = H_2(Y^{H_1(r_A,a)})$.

IV) The shared session key is K.

The protocol is however insecure if the ephemeral keys are defined to contain the

[2] http://eprint.iacr.org/cgi-bin/versions.pl?entry=2009/253, version 20090625.

ephemeral DH exponents. As an adversary which (*partially*[3]) learns $H_1(r_A, a)$ in a session initiated at $\hat{A}$ with peer $\hat{B}$, can *indefinitely* impersonate $\hat{A}$ to $\hat{B}$. For this purpose, the attacker replays to $\hat{B}$ $\hat{A}$'s message in the session in which $H_1(r_A, a)$ leakage happened (namely $(\hat{B}, X, \sigma_A)$), and computes the session key that $\hat{B}$ derives, using $H_1(r_A, a)$ and the ephemeral public key Y from $\hat{B}$.

3 Stronger Security

In this section, we describe the strengthened eCK model, which considers leakages on intermediate results (the values a party may need to compute between messages or before a session key), encompasses the eCK model [17], and provides stronger reveal queries to the attacker.

A common setting wherein key agreement protocols are often implemented is that of a server used together with a (computationally limited) tamper–resistant device, which stores the long–lived secrets. In such a setting, safely reducing the non–idle time computational effort of the device, is usually crucial for implementation efficiency. To reduce the device's non–idle time computational effort, ephemeral keys can be computed on the device in idle–time, or on the host machine when the implemented protocol is ephemeral DH exponent leakage resilient.

In many DH protocols, (C, FH, H)MQV–C [18, 33, 29, 14, 15] and NAXOS(+,–C) [20, 24, 17], for instance, the computation of the intermediate results is more costly than that of the ephemeral public key. For these protocols, implementations efficiency is significantly enhanced when the ephemeral keys are computed on the device, while the intermediate results, which require expensive on–line computations and session keys are computed on the host machine. Unfortunately the security of the (e)CK–secure protocols, when leakages on the intermediate results are considered is at best unspecified. A security definition which captures attacks based on intermediate result leakages is clearly desirable. The model we propose captures such attacks, together with the attacks captured in the (e)CK models.

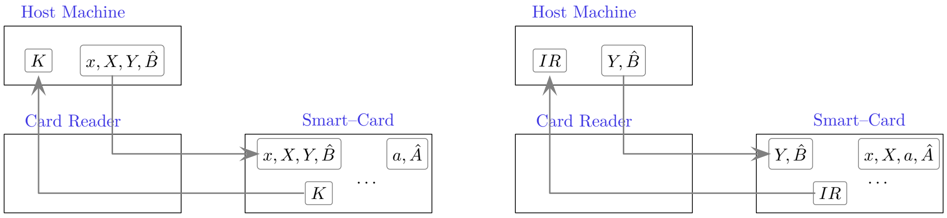
Session. We suppose $n \leqslant \mathcal{L}(|q|)$ (for some polynomial $\mathcal{L}$) parties $\hat{P}_{i=1,\cdots,n}$ supposed to be probabilistic polynomial time machines and a certification authority (CA) trusted by all parties. The CA is only required to verify that public keys are valid ones (i.e., public keys are only tested for membership in $\mathcal{G}^*$; no proof of possession of corresponding private keys is required). Each party has a certificate binding its identity to its public key. A session is an instance of the considered protocol, run at a party. A session at $\hat{A}$ (with peer $\hat{B}$) can be created with parameter $(\hat{A}, \hat{B})$ or $(\hat{B}, \hat{A}, m)$, where m is an incoming message, supposed from $\hat{B}$; $\hat{A}$ is the initiator if the creation parameter is $(\hat{A}, \hat{B})$, otherwise a responder. At session activation, a session state is created to contain the information specific to the session. Each session is identified with a tuple $(\hat{P}_i, \hat{P}_j, \mathfrak{out}, \mathfrak{in}, \varsigma)$, wherein $\hat{P}_i$ is the session holder, $\hat{P}_j$ is the intended peer, $\mathfrak{out}$ and $\mathfrak{in}$ are respectively the concatenation of the messages $\hat{P}_i$ sends to $\hat{P}_j$, or believes to be from $\hat{P}_j$, and ς is $\hat{P}_i$'s role in the session (initiator or responder). Two sessions with identifiers $(\hat{P}_i, \hat{P}_j, \mathfrak{out}, \mathfrak{in}, \varsigma)$ and $(\hat{P}'_j, \hat{P}'_i, \mathfrak{out}', \mathfrak{in}'\varsigma')$ are said to be matching if $\hat{P}_i = \hat{P}'_i$, $\hat{P}'_j = \hat{P}'_j$, $\varsigma \neq \varsigma'$, and at completion $\mathfrak{in} = \mathfrak{out}'$ and $\mathfrak{out} = \mathfrak{in}'$.

[3] If an adversary partially learns $H_1(r_A, a)$, it recovers the remaining part, using Shanks' baby step giant step algorithm [32] or Pollard's rho algorithm [32], if the bits it learns are the most significant ones, or tools from [11] if the leakage is on middle–part bits; recovering $H_1(r_A, a)$ from partial leakage requires some extra computational effort.

For the two–pass DH protocols, each session is denoted with an identifier $(\hat{A}, \hat{B}, X, \star, \varsigma)$, where $\hat{A}$ is the session holder, $\hat{B}$ is the peer, X is the outgoing message, ς indicates the role of $\hat{A}$ in the session (initiator $(\mathcal{I})$ or responder $(\mathcal{R})$), and $\star$ is the incoming message Y if it exists, otherwise a special symbol meaning that an incoming message is not received yet; in that case, when $\hat{A}$ receives the public key Y, the session identifier is updated to $(\hat{A}, \hat{B}, X, Y, \varsigma)$. Two sessions with identifiers $(\hat{A}, \hat{B}, X, Y, \mathcal{I})$ and $(\hat{B}, \hat{A}, Y, X, \mathcal{R})$ are said to be matching. Notice that the session matching $(\hat{B}, \hat{A}, Y, X, \mathcal{R})$ can be any session $(\hat{A}, \hat{B}, X, \star, \mathcal{I})$; as $X, Y \in_R \mathcal{G}^*$, a session cannot have (except with negligible probability) more than one matching session.

Adversary and Security. The adversary $\mathcal{A}$, is a probabilistic polynomial time machine; outgoing messages are submitted to $\mathcal{A}$ for delivery ($\mathcal{A}$ decides about messages delivery). $\mathcal{A}$ is also supposed to control session activations at each party via the *Send*$(\hat{P}_i, \hat{P}_j)$ and *Send*$(\hat{P}_j, \hat{P}_i, Y)$ queries, which make $\hat{P}_i$ initiate a session with peer $\hat{P}_j$, or respond to the (supposed) session $(\hat{P}_j, \hat{P}_i, Y, \star, \mathcal{I})$. We suppose that the considered protocol is implemented at a party following one of the approaches hereunder. We suppose also that at each party an untrusted host machine is used together with a tamper–resistant device. Basing our model on these implementation approaches does not make it specific; rather, this reduces the gap that often exists between formal models and practical security. Such modeling techniques, which take into account hardware devices and communication flows between components, were previously used in [6].

Approach 1. In this approach, the static keys are stored on the device (a smart–card, for instance) the ephemeral keys are computed on the host machine, passed to the smart–card together with the incoming public keys; the device computes the session key, and provides it to the host machine (application) for use. The information flow between the device and the host machine is depicted in Figure (1a). This implementation approach is safe for eCK–secure protocols when ephemeral keys are defined to be ephemeral DH exponents, as a leakage on an ephemeral DH exponent does not compromise the session in which it is used. In addition, when an attacker learns a session key, it gains no useful information about the other session keys.



(a) Implementation Approach 1

(b) Implementation Approach 2

Figure 1: Implementation Approaches

Approach 2. Another approach, which has received less attention in the formal treatment of DH protocols, is when the ephemeral keys, and top level intermediate results are computed on the device, and the host machine is provided with some intermediate results IR with which it computes the session key. As the computation of the intermediate results is often more costly than that of the ephemeral public keys, implementation efficiency is often significantly enhanced using this approach. Naturally, this comes with the requirement that leakages on the intermediate results should not compromise any unexposed session. Namely, an adversary may have a malware running on the host machine at a party, and learn all values computed or used at the party, except those stored in the party's tamper–proof device; this should not compromise any unexposed session.

We define two sets of queries, modeling leakages that may occur on either implementation approaches. We consider leakages on ephemeral and static private keys, and also on any intermediate (secret) value which evaluation requires a secret information. As the adversary can compute any information which evaluation requires only public information, considering leakages on such data is superfluous.

In Set 1, which models leakages in the first implementation approach, the following queries are allowed.

- *EphemeralKeyReveal(session)*: this query models leakages on ephemeral DH exponents.
- *Corrupt$_{SC}$(party)*: this query models an attacker which (bypasses the eventual tamper protection mechanisms on the device, and) gains read access to the device's private memory; it provides the attacker with the device owner's static private key.
- *SessionKeyReveal(session)*: when the attacker issues this query on an already completed session, it is provided with the session key.
- *EstablishParty(party)*: with this query, the adversary registers a static key on behalf of a party; as the adversary controls communications, from there the party is supposed totally controlled by $\mathcal{A}$. A party against which this query is not issued is said to be *honest.*

In Set 2, which models leakages on the second implementation approach, the following queries are allowed; the definitions remain unchanged for the queries belonging also to Set 1.

- For any node in the *intermediate results,* which computation requires a secret value, a reveal query is defined to allow leakage on the information computed in this node. These queries models leakages that may occur on intermediate results in computing session keys.
- *SessionKeyReveal(session).*
- *EstablishParty(party).*
- *Corrupt$_{SC}$(party).*

Before defining the seCK security, we define the session freshness notion. Test queries can only be performed on fresh sessions.

Definition 1 (Session Freshness)**.** Let Π be a protocol, and $\hat{A}$ and $\hat{B}$ two honest parties, *sid* the identifier of a completed session at $\hat{A}$ with peer $\hat{B}$, and *sid′* the matching session's identifier. The session *sid* is said to be *locally exposed* if one of the following holds.

- $\mathcal{A}$ issues a *SessionKeyReveal* query on *sid.*
- The implementation at $\hat{A}$ follows the first approach and $\mathcal{A}$ issues an *EphemeralKeyReveal* query on *sid* and a *Corrupt$_{SC}$* query on $\hat{A}$.
- The implementation at $\hat{A}$ follows the second approach and $\mathcal{A}$ issues an intermediate result query on *sid.*

The session *sid* is said to be *exposed* if (a) it is locally exposed, or (b) its matching session *sid′* exists and is locally exposed, or (c) *sid′* does not exist and $\mathcal{A}$ issues a *Corrupt$_{SC}$* query on the supposed peer $\hat{B}$. An *unexposed* session is said to be *fresh.*

Our session freshness conditions match exactly the intuition of the sessions one may hope to protect. In particular, it lowers (more than in the eCK model) the necessary adversary restrictions for any reasonable security definition. Notice that only the queries corresponding to the implementation approach followed by a party can be issued on it.

Definition 2 (Strengthened eCK Security)**.** Let Π be a protocol, such that if two honest parties complete matching sessions, then they both compute the same session key.

The protocol Π is said to be *seCK–secure*, if no polynomially bounded adversary can distinguish a fresh session key from a random value, chosen under the distribution of session keys, with probability significantly greater than $1/2$.

Forward Secrecy. As shown in [14], no implicitely authenticated two–pass key exchange protocol can achieve *forward secrecy*[4]. Indeed, our security definition captures *weak forward secrecy*, which (loosely speaking) is: *any session established without an active involvement of the attacker remains secure, even when the implicated parties static keys are disclosed.* The seCK security definition can be completed with the session key expiration notion [7] to capture forward secrecy. Although the protocol we propose can be added with a third message, and yield a protocol which (provably) provides forward secrecy, in the continuation, we work with the security definition without forward secrecy, and focus on two–pass DH protocols.

Relations between the seCK and eCK models. In the eCK model, an adversary may compromise the ephemeral key, static key, or the session key at a party, independently of the way the protocol is implemented. The seCK model considers an adversary which may (have a malware running at a party's host machine and) learn all information at the party, except those stored in a tamper–resistant device. The seCK approach seems more prevalent in practice, and reduces the gap that often exists between formal arguments and practical implementations security.

The eCK and seCK session identifiers and matching sessions definitions are the same. When the adversary issues the $Corrupt_{SC}$ query at a party, it is provided with the party's static key; the $Corrupt_{SC}$ query is the same as the eCK *StaticKeyReveal* query. For a session between two parties, say $\hat{A}$ and $\hat{B}$, following the first implementation approach, the seCK session *freshness* definition reduces to the eCK freshness. By assuming that all parties follow the first implementation approach, the seCK–security definition reduces to the eCK one; the seCK model encompasses the eCK one.

Proposition 1. *Any seCK–secure protocol is also an eCK–secure one.*

The seCK model also separates clearly from the eCK model. The eCK model does not consider leakages on intermediate results; and this makes many of the eCK secure protocols insecure in the seCK model. For instance, in the CMQV protocol (shown eCK–secure), an attacker which learns an ephemeral secret exponent in a session, can indefinitely impersonate the session owner; the same holds for the (H)MQV(–C) protocols [29, 30]. It is not difficult to see that NAXOS cannot meet the seCK security definition. The protocols 1 and 2 from [12, pp. 6, 12] (shown eCK–secure) fail in authentication when leakages one the intermediate results are considered. Indeed an attacker, which learns the ephemeral secret exponents $s_1 = x + a_1$ and $s_2 = x + a_2$ in a session at $\hat{A}$ (see the steps 2 and 3 of the protocols 1 and 2 [12]), can indefinitely impersonate $\hat{A}$ to *any* party. Notice that the attacker cannot compute $\hat{A}$'s static key from s_1 and s_2, while it is not difficult to see that leakages on s_1 (or s_2) *and* the ephemeral key, in the *same session* imply $\hat{A}$'s static key disclosure.

The seCK model is practically stronger than the CK model [7]. Key Compromise Impersonation resilience, for instance, is captured in the seCK model while not in CK model. As shown in [9], and illustrated in section 2 with Protocol $\mathcal{P}$, the CK model is enhanced when matching sessions are defined using matching conversations. In addition, the seCK

[4] Some authors, [14] for instance, use the term 'perfect forward secrecy', but we prefer 'forward secrecy' to avoid a confusion with (Shannon's) 'perfect secrecy'.

reveal query definitions go beyond the usual CK session state definition (ephemeral DH exponents). Compared to the CK_{HMQV} model[5] [14], the reveal query definitions are enhanced in the seCK model to capture attacks based on intermediate result leakages. In the HMQV security arguments [14, subsection 7.4], the session state is defined to contain the ephemeral DH exponent[6]; the HMQV protocol does not meet the seCK–security [29, 30].

4 The Strengthened MQV Protocol

In this section, we present the *strengthened* MQV protocol, and its building blocks, to show that the seCK security definition is useful, and not limiting; as seCK–secure protocols can be built with usual building blocks. We start with the following variants of the FXCR and FDCR signature schemes [29]. The security of the FXCR–1 and FDCR–1 schemes can be shown with arguments similar to that of the FXCR and FDCR schemes [29, 30].

Definition 3 (FXCR–1 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 $\hat{A}$) 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 $s_B = ye + b$, where $e = \bar{H}(Y, X, m)$. And, $\hat{A}$ accepts the pair (Y, σ_B) as a valid signature if $Y \in \mathcal{G}^*$ and $(Y^e B)^x = \sigma_B$.

Proposition 2 (FXCR–1 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 hashing and 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).

Definition 4 (FDCR–1 Scheme)**.** Let $\hat{A}$ and $\hat{B}$ be two parties with public keys $A, B \in \mathcal{G}^*$, and m_1, m_2 two messages. The dual signature of $\hat{A}$ and $\hat{B}$ on the messages m_1, m_2 is $DSig_{\hat{A},\hat{B}}(m_1, m_2, X, Y) = (X^d A)^{ye+b} = (Y^e B)^{xd+a}$, where $X = G^x$ and $Y = G^y$ are chosen respectively by $\hat{A}$ and $\hat{B}$, $d = \bar{H}(X, Y, m_1, m_2)$, and $e = \bar{H}(Y, X, m_1, m_2)$.

Proposition 3 (FDCR–1 Security)**.** *Let $A = G^a, B, X_0 \in_R \mathcal{G}^*$ ($A \neq B$). Under the RO model, and the CDH assumption in $\mathcal{G}$, given a, A, B, X_0, a message m_{1_0}, a hashing oracle, together with a signing oracle (simulating $\hat{B}$'s role), no adaptive probabilistic polynomial time attacker can output, with non–negligible success probability a triple (m_{2_0}, Y_0, σ_0) such that:*

(1) $DSig_{\hat{A},\hat{B}}(m_{1_0}, m_{2_0}, X_0, Y_0) = \sigma_0$; and

(2) (Y_0, σ_0) was not obtained from the signing oracle with a query on some (m'_1, X') such that $X_0 = X'$ and $(m'_1, m'_2) = (m_{1_0}, m_{2_0})$, where m'_2 is a message returned at signature query on (m'_1, X'); (m_{1_0}, m_{2_0}) denotes the concatenation of m_{1_0} and m_{2_0}.

The strengthened MQV protocol follows from the FDCR–1 scheme; a run of SMQV is as in Protocol 3. The execution aborts if any verification fails.

[5] CK_{HMQV} is the variant of the CK model in which the HMQV security arguments are provided; however, it seems that the aim of [14] was not to propose a new model, as it refers to [7] for details [14, p. 9], and considers its session identifiers and matching sessions definition (which make the CK and CK_{HMQV} models incomparable) as consistent with the CK model [14, p. 10]. See [10] for a comparison between the CK_{HMQV} and (e)CK models.

[6] In [14, subsection 5.1], the session state is defined to contain the ephemeral public keys, but this definition is superfluous, as the adversary controls communications between parties.

Protocol 3 The Strengthened MQV Protocol

I) The initiator $\hat{A}$ does the following:
   (a) Choose $x \in_R [1, q-1]$ and compute $X = G^x$.
   (b) Send $(\hat{A}, \hat{B}, X)$ to the peer $\hat{B}$.
II) At receipt of $(\hat{A}, \hat{B}, X)$, $\hat{B}$ 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 $(\hat{B}, \hat{A}, Y)$ to $\hat{A}$.
   (d) Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B})$ and $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.
   (e) Compute $s_B = ye + b \mod q$ and $\sigma = (X^d A)^{s_B}$.
   (f) Compute $K = H(\sigma, \hat{A}, \hat{B}, X, Y)$.
III) At receipt of $(\hat{B}, \hat{A}, Y)$, $\hat{A}$ 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 $s_A = xd + a \mod q$, and $\sigma = (Y^e B)^{s_A}$.
   (d) Compute $K = H(\sigma, \hat{A}, \hat{B}, X, Y)$.
IV) The shared session key is K.

Remark 1. It may appear that the the SMQV and FHMQV protocols are the same, but this is not the case. Indeed, when implementing the FHMQV or SMQV protocol, it may be desirable to reduce the sensitivity of implementation to side channel attacks [13]. In the second implementation approach, this seems to be more convenient in SMQV than in FHMQV. In fact, in the FHMQV protocol, the static private keys are always multiplied with publicly known digest values, and then potentially sensitive to well–targeted side channel attacks. In the SMQV protocol, the static private keys are only used in an addition, with the second operand (involving the ephemeral private key) used once, and unknown to the attacker. This makes side channel attacks aiming to reveal the static private keys significantly more difficult against SMQV than against FHMQV.

In SMQV, the shared secret σ is the FDCR–1 signature of $\hat{A}$ and $\hat{B}$, on challenges X, Y and messages $\hat{A}, \hat{B}$ (the representation of $\hat{A}$ and $\hat{B}$'s identities). The parties identities and ephemeral keys are used in the final digest computation to make the key replication resilience security attribute immediate (and also to avoid unknown key share attacks). A run of SMQV requires 2.5 times a single exponentiation (2.17 times a single exponentiation when the multiple exponentiation technique [22, Algorithm 14.88] is used); this efficiency equals that of the remarkable (H, FH)MQV protocols. SMQV provides all the security attributes of the (C, H)MQV protocols, added with ephemeral secret exponent leakage resilience.

Moreover, suppose an implementation of SMQV or (C, H)MQV using an untrusted[7] host machine together with a device; and suppose that the session keys are used by some applications running on the host machine, and that the ephemeral keys are computed on the device in idle–time. This idle–time pre–computation seems common in practice [28] (and possible in both the (C, H)MQV and SMQV protocols). But, as (C, H)MQV is not ephemeral secret exponent leakage resilient [29, 30], the ephemeral secret exponents ($s_A = x + da$ or $s_B = y + eb$) cannot be used on the untrusted host machine. The exponentiation $\sigma = (YB^e)^{s_A} = (XA^d)^{s_B}$ has to be performed on the device *in non idle–time*. In contrast, for SMQV, $\sigma = (Y^e B)^{s_A} = (X^d A)^{s_B}$ can be computed on the host

[7] There are many reasons for not trusting the host machine: bogus or trojan softwares, viruses, etc.

machine, after the ephemeral secret exponent is computed on the device. Because the session key is used on the host machine, and a leakage of only the ephemeral secret exponent, in a SMQV session, does not compromise any other session; there is no need to protect the ephemeral secret exponent more than the session key. In SMQV, the non–idle time computational effort of the device reduces to few non–costly operations (one integer addition, one integer multiplication, and one digest computation), while for (C, H)MQV at least one exponentiation has to be performed on the device in non idle–time.

Table 1: Security and Efficiency Comparison between SMQV and other DH protocols.

Protocol	Security	Assumptions	NC	NICE 1	NICE 2
CMQV [33]	eCK	GDH	3E	1E	1E
FHMQV [30]	CK_{FHMQV}	GDH	2.5E	1E	1D + 1A + 1M
HMQV [14]	CK_{HMQV}	GDH, KEA1	2.5E	1E	1E
MQV [18]	–	–	2.5E	1E	1E
NAXOS [17]	eCK	GDH	4E	3E	3E
NAXOS–C [24]	ceCK	GDH	4E	3E	3E
SMQV	seCK	GDH	2.5E	1E	1D + 1A + 1M

Table 1 summarizes the comparisons between SMQV and some other DH protocols. All the security reductions are performed using the Random Oracle model [2]; incoming ephemeral keys are validated[8]. KEA1 stands for “Knowledge of Exponent Assumption” [4], CDH and GDH stand respectively for “Computational DH” and “Gap DH” assumptions [26]. The *‘A’, ‘D’, ‘E’*, and *‘M’* stand respectively for *integer* a*ddition,* d*igest computation,* e*xponentiation, and integer* m*ultiplication.* The NC column indicates the *n*aive *c*ount efficiency (i.e., without optimizations from [22, Algorithm 14.88] and [25]); NICE 1 and NICE 2 indicate the *n*on–*i*dle time *c*omputational *e*ffort of the device in the two approaches (when ephemeral keys are computed in idle–time).

The MQV protocol has no security reduction[9]. The FHMQV security arguments are provided in a model which considers intermediate results and ephemeral key leakages in two separate settings; the model implicitly assumes that all parties follow the same implementation approach, and cannot be shown to encompass the CK or eCK models. In contrast, the seCK model considers also the security of sessions between parties following different implementation approaches, and its matching sessions definition makes it encompass the eCK model. The CMQV and NAXOS protocols are shown eCK–secure, they both use the NAXOS transformation.

The NAXOS–C security arguments are provided in a variant of the eCK model, called *combined eCK model* (ceCK) [24], geared to the post–specified peer model. In the post[10] model, the identity of a peer may be unknown at session activation (it is learned during the protocol execution). It is worthwhile to mention that, the separation between the pre and post models security seems unclear. The protocol $\mathcal{P}$ claimed secure in the pre model, and not executable in the post model (unless “modified in a fundamental way”) [24, section 3.1], is insecure in the pre model, if the considered security model is strong enough.

[8] Ephemeral key validation is voluntarily omitted in the HMQV design [14], but the HMQV protocol is known to be insecure if ephemeral keys are not validated [23].

[9] We are aware of [16], which shows that under the RO model and the CDH assumption, the MQV variant wherein d and e are computed as $\bar{H}(X)$ and $\bar{H}(Y)$, is secure in a model of their own design. But, for this MQV variant, an attacker which finds $x_0 \in [1, q-1]$ such that $\bar{H}(G^{x_0}) = 0$, can impersonate *any* party to *any other* party. Finding such an x_0 requires $O(2^l)$ digest computations.

[10] The terms ‘pre–specified peer’ and ‘post–specified peer’ are respectively shortened to ‘pre’ and ‘post’.

The HMQV protocol is executable in the post model, but claimed insecure (in the post–model). In fact, the proposed attack [24, section 3.2] cannot be performed in practice; not because it requires an important on–line computational effort (2^{60} operations, when the order of $\mathcal{G}$ is a 160–bit prime), but since the step (2.c) of the attack cannot be performed without changing the $\hat{M}$ found at the step (2.b). In practice, $\hat{M}$ (is a certificate, and) is defined to contain M (which is provided to the certification authority at certificate issuance), and when M is changed, so is $\hat{M}$ (notice also that changing M requires another certificate issuance); and then, after the step (2.c) of the attack, the claimed equality between $\bar{H}(X, \hat{M})$ and $\bar{H}(X, \hat{B})$ does not hold. For the Σ_0 protocol, secure in the post model, while insecure in the pre one [24, section 3.3], the model in which it is shown secure in the post model [8] is not strong enough. It is not difficult to see, for instance, that the Σ_0 protocol is both eCK and ceCK insecure.

The SMQV protocol provides more security attributes than the NAXOS(+, –C), (C, H)MVQ protocols, in addition to allow particularly efficient implementations, in environments wherein a tamper proof device is used to store private keys.

Proposition 4. *Under the GDH assumption in $\mathcal{G}$ and the RO model, the SMQV protocol is seCK–secure.*

5 Concluding Remarks

We discussed security shades in the (e)CK moddels. We illustrated the limitations of the CK matching sessions definition; and the insecurity of the NAXOS type protocols when leakages on ephemeral DH exponents are considered. We proposed a new security model, the strengthened eCK model, which encompasses the eCK one, and practically captures the security attributes considered in the CK model. We proposed the Strengthened MQV protocol, which in addition to provide the same efficiency as the (H)MQV protocols, is particularly suited for distributed implementation environments using an untrusted host machine and a tamper–resistant device; in such an environment, the non–idle time computational effort of the device, in a SMQV implementation, reduces to few non–costly operations.

In a forthcoming stage, we will be interested in the enhancement of existing protocols to meet the seCK security definition, and the extension of the strengthened eCK model to consider a wider class of attacks.

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] Bellare M., Rogaway P.: Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In Proc. of the first ACM Conference on Computer and Communications Security, pp. 62–73, ACM, 1993.

[3] Bellare M., Rogaway P.: Provably Secure Session Key Distribution — The Three Party Case. In Proc. of the twenty–seventh annual ACM symposium on Theory of computing, pp. 57–66, ACM, 1995.

[4] Bellare M., Palacio A.: The Knowledge–of–Exponent Assumptions and 3–round Zero–Knowledge Protocols. In Proc. of Crypto 04, Lecture Notes in Computer Science, vol. 3152, 273–289, Springer–Verlag, 2004.

[5] Blake–Wilson S., Johnson D., Menezes A.: Key Agreement Protocols and their Security Analysis. In Proc of the sixth IMA International Conference on Cryptography and Coding, Lecture Notes of Computer Science, vol. 1355, pp. 30–45, Springer–Verlag, 1997.

[6] Bresson E., Chevassut O., Pointcheval D.: Dynamic Group Diffie–Hellman Key Exchange under Standard Assumptions. In Proc. of Eurocrypt 02, Lecture Notes in Computer Science, vol. 2332, pp. 321–336, Springer–Verlag, 2002.

[7] 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.

[8] Canetti R., Krawczyk H.: Security Analysis of IKE's Signature–based Key–Exchange Protocol. In Proc of Crypto 02, Lecture Notes in Computer Science, vol. 2442, pp. 143–161. Springer–Verlag, 2002.

[9] Choo K.–K. R., Boyd C., Hitchcock Y.: Examining Indistinguishability–Based Proof Models for Key Establishment Protocols. In Proc. of Asiacrypt 05, Lecture Notes in Computer Science, vol 3788, pp. 585-604, Springer–Verlag, 2005.

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

[11] Gopalakrishnan K., Thériault N., Yao C. Z.: Solving Discrete Logarithms from Partial Knowledge of the Key. In Proc. of Indocrypt 07, Lecture Notes in Computer Science, vol. 4859, pp. 224–237, Springer–Verlag, 2007.

[12] Kim M., Fujioka A., Ustaoglu B.: Strongly Secure Authenticated Key Exchange without NAXOS' Approach. In Proc. of the fourth International Workshop on Security, IWSEC 09, Lecture Notes in Computer Science, vol. 5824, pp. 174–191, Springer–Verlag, 2009.

[13] Kocher P.: Timing attacks on implementations of Diffe–Hellman, RSA, DSS, and other systems. Advances in Cryptology – Crypto 96, Lecture Notes in Computer Science, vol. 1109, pp. 104–113, Springer–Verlag, 1996.

[14] Krawczyk H.: HMQV: A Hight Performance Secure Diffie–Hellman Protocol. Cryptology ePrint Archive, Report 2005/176, 2005.

[15] 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.

[16] Kunz-Jacques S., Pointcheval D.: About the Security of MTI/C0 and MQV. In proc. of the international conference Security and Cryptography for Networks, SCN 2006, Lecture Notes in Computer Science vol. 4116, pp. 156–172, Springer–Verlag, 2006.

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

[18] Law L., Menezes A., Qu M., Solinas J., Vanstone S.: *An Efficient protocol for authenticated key agreement,* Designs, Codes and Cryptography, vol. 28(2), pp. 119–134, Kluwer Academic Publishers, 2003.

[19] J. Lee, C. S. Park. *An Efficient Authenticated Key Exchange Protocol with a Tight Security Reduction.* Cryptology ePrint Archive, Report 2008/345, 2008.

[20] Lee J., Park J. H.: Authenticated Key Exchange Secure under the Computational Diffe–Hellman Assumption. Cryptology ePrint Archive, Report 2008/344, 2008.

[21] 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.

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

[23] Menezes A., Ustaoglu B.: On the Importance of Public–Key Validation in the MQV and HMQV Key Agreement Protocols. In Proc. of Indocrypt 06, Lecture Notes in Computer Science, vol. 4329, pp. 133-147, Springer–Verlag, 2006.

[24] 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.

[25] M'Raïhi D., Naccache D.: Batch Exponentiation: A Fast DLP-based Signature Generation Strategy. In Proc. of the third ACM conference on Computer and communications security, pp. 58–61, ACM, 1996.

[26] Okamoto T., Pointcheval D.: The Gap–Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Proc. of Crypto 96, Lecture Notes in Computer Science, vol. 1992, pp. 104–118, Springer–Verlag, 2001.

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

[28] Schnorr C. P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology, vol. 4(3), pp. 161–174, Springer New York, 1991.

[29] Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A Secure and Efficient Authenticated Diffie–Hellman Protocol. To appear in Proc. of EuroPKI'09, 2009.

[30] Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A Secure and Efficient Authenticated Diffie–Hellman Protocol (extended version). Cryptology ePrint Archive, Report 2009/408, 2009.

[31] Shoup V.: On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, 1999/012, 1999.

[32] Teske E.: Square–root Algorithms for the Discrete Logarithm Problem (A survey). Public Key Cryptography and Computational Number Theory, pp. 283–301, Walter de Gruyter, 2001

[33] Ustaoglu B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography, vol. 46(3), pp. 329–342, Kluwer Academic Publishers, 2008.

[34] Ustaoglu B.: Comparing SessionStateReveal and EphemeralKeyReveal for Diffe–Hellman protocols. In Proc. of Provable Security 2009, Lecture Notes in Computer Science, vol. 5848, pp. 183–197, Springer–Verlag, 2009.

A Security Analysis of the SMQV Protocol

In accordance with our security model, the following session activation queries are allowed.

- *Send*$(\hat{A}, \hat{B})$*,* which makes $\hat{A}$ perform the step I) of Protocol 3, and create a session with identifier $(\hat{A}, \hat{B}, X, \star, \mathcal{I})$.
- *Send*$(\hat{A}, \hat{B}, X)$*,* which makes $\hat{B}$ perform the step II) of Protocol 3, and create a session with identifier $(\hat{B}, \hat{A}, Y, X, \mathcal{R})$.
- *Send*$(\hat{A}, \hat{B}, X, Y)$, which makes $\hat{A}$ update the session identifier $(\hat{A}, \hat{B}, X, \star, \mathcal{I})$ (if any) to $(\hat{A}, \hat{B}, X, Y, \mathcal{I})$ and perform the step III) of SMQV.

The queries in Set 1 are the following: *EphemeralKeyReveal, Corrupt*$_{SC}$*, SessionKeyReveal,* and *EstablishParty.* In Set 2, the allowed queries are: (*i*) *Corrupt*$_{SC}$, to obtain the static private key of a party; (*ii*) *SessionKeyReveal*, to obtain a session key; (*iii*) *SecretExponentReveal*, to obtain a secret exponent $s = xd+a$ or $ye+b$; (*iv*) *SessionSignatureReveal*, to obtain a session signature σ; (*v*) *EstablishParty(party)* to register a static public key on behalf of a party.

Recall that an algorithm is said to be a *Decisional Diffie–Hellman Oracle* (DDHO) if on input $G, X = G^x, Y = G^y$, and $Z \in_R \mathcal{G}$, it outputs 1 if and only if $Z = G^{xy}$. And the Gap DH (GDH) assumption [26] is said to hold in $\mathcal{G}^*$ if given a DDHO, there is no polynomially bounded algorithm, which solves the CDH problem in $\mathcal{G}$, with non–negligible success probability.

A.1 Proof of Proposition 4.

It is immediate from the definition of SMQV that if two honest parties complete matching sessions, they compute the same session key. Suppose an attacker $\mathcal{A}$, which succeeds with probability significantly greater than 1/2 in distinguishing a fresh session key from a random value chosen under the distribution of session keys. Distinguishing a fresh session key from a random value can be performed only in one of the following ways.

Guessing attack: $\mathcal{A}$ guesses correctly the test session key.

Key replication attack: $\mathcal{A}$ succeeds in making two non–matching sessions yield the same session key, it then issues a session key reveal query on one of the sessions, and uses the other as test session.

Forging attack: $\mathcal{A}$ computes the session signature σ, and issues a digest query to get the session key.

Under the RO model, guessing and key replications attacks cannot succeed, except with negligible probability. Key replication attacks cannot succeed, as if $X \neq X'$, or $Y \neq Y'$, or $\hat{A} \neq \hat{A}'$, or $\hat{B} \neq \hat{B}'$, and no substring of $\hat{A}$ equals $\hat{B}$ (and conversely) the probability that $H(\sigma, \hat{A}, \hat{B}, X, Y)$ equals $H(\sigma', \hat{A}', \hat{B}', X', Y')$ is negligible. We thus suppose that $\mathcal{A}$ succeeds with non–negligible probability in forging attack. Let E be the event "$\mathcal{A}$ succeeds in forging the signature of some fresh session (that we designate by $sid_0 = (\hat{A}, \hat{B}, X_0, Y_0, \varsigma)$)." The event E divides in $\mathsf{E.1}$: "$\mathcal{A}$ succeeds in forging the signature of a fresh with matching session," and $\mathsf{E.2}$: "$\mathcal{A}$ succeeds in forging the signature of a fresh without matching session." It suffices to show that neither $\mathsf{E.1}$ nor $\mathsf{E.2}$ can happen with non–negligible[11] probability.

Analysis of $\mathsf{E.1}$

Suppose that $\mathsf{E.1}$ occurs with non–negligible probability; at least one of the following events occurs with non–negligible probability.

$\mathsf{E.1.1}$: "$\mathsf{E.1} \wedge$ both $\hat{A}$ and $\hat{B}$ follow the first implementation approach";

$\mathsf{E.1.2}$: "$\mathsf{E.1} \wedge$ both $\hat{A}$ and $\hat{B}$ follow the second implementation approach";

$\mathsf{E.1.3}$: "$\mathsf{E.1} \wedge$ $\hat{A}$ and $\hat{B}$ follow different implementation approaches."

We have to show that none of $\mathsf{E.1.1}$, $\mathsf{E.1.2}$ and $\mathsf{E.1.3}$ can occur, except with negligible probability.

Analysis of $\mathsf{E.1.1}$. Since the test session is required to be fresh, the strongest queries that $\mathcal{A}$ can perform on $\hat{A}$, $\hat{B}$, the test session, and its matching session are (*i*) $Corrupt_{SC}$ queries on both $\hat{A}$ and $\hat{B}$; (*ii*) *EphemeralKeyReveal* queries on both sid_0 and sid_0'; (*iii*) a $Corrupt_{SC}$ query on $\hat{A}$ and an *EphemeralKeyReveal* query on sid_0'; (*iv*) an *EphemeralKeyReveal* query on sid_0 and a $Corrupt_{SC}$ query on $\hat{B}$. It thus suffices to show that none of the following events can happen with non–negligible probability since from any polynomial time machine, which succeeds in $\mathsf{E.1.1}$ and performs weaker queries, one can build a polynomial time machine which succeeds with the same probability, and performs one the strongest allowed queries.

$\mathsf{E.1.1.1}$: "$\mathsf{E.1.1} \wedge \mathcal{A}$ issues $Corrupt_{SC}$ queries on both $\hat{A}$ and $\hat{B}$";

$\mathsf{E.1.1.2}$: "$\mathsf{E.1.1} \wedge \mathcal{A}$ issues *EphemeralKeyReveal* queries on both sid_0 and sid_0'";

$\mathsf{E.1.1.3}$: "$\mathsf{E.1.1} \wedge \mathcal{A}$ issues a $Corrupt_{SC}$ query on $\hat{A}$ and an *EphemeralKeyReveal* query on sid_0'";

$\mathsf{E.1.1.4}$: "$\mathsf{E.1.1} \wedge \mathcal{A}$ issues an *EphemeralKeyReveal* query on sid_0 and a $Corrupt_{SC}$ query on $\hat{B}$."

Event $\mathsf{E.1.1.1}$. Suppose that $\mathsf{E.1.1.1}$ occurs with non–negligible probability, using $\mathcal{A}$ we build a polynomial time CDH solver $\mathcal{S}$, which succeeds with non–negligible probability.

[11] A function $\mathcal{F}$ with parameter ξ is said to be negligible, if for every polynomial $\mathcal{L}$, and every sufficiently large ξ, $\mathcal{F}(\xi) < (|\mathcal{L}(\xi)|)^{-1}$; otherwise $\mathcal{F}$ is said to be non–negligible.

The solver interacts with $\mathcal{A}$ as follows.

(1) $\mathcal{S}$ simulates $\mathcal{A}$'s environment, with n parties $\hat{P}_1, \ldots, \hat{P}_n$, and assigns to each $\hat{P}_k$ a random static key pair $(p_k, P_k = G^{p_k})$, together with an implementation approach indication. We only suppose that the number of parties following the first implementation approach is $n_1 \geqslant 2$. $\mathcal{S}$ starts with two empty digest records $\mathfrak{H}_1$ and $\mathfrak{H}_2$. Since $\mathcal{A}$ is polynomial (in $|q|$), we suppose that each party is activated at most m times ($m, n \leqslant \mathcal{L}(|q|)$ for some polynomial $\mathcal{L}$). $\mathcal{S}$ chooses $i, j \in_R \{k \mid \hat{P}_k$ follows the first implementation approach$\}$, and $t \in_R [1, m]$ (with these choices, $\mathcal{S}$ is guessing the test session). We refer to $\hat{P}_i$ as $\hat{A}$ and $\hat{P}_j$ as $\hat{B}$.

(2) At $\bar{H}$ digest query on some $\varrho = (X, Y, \hat{P}_l, \hat{P}_m)$, $\mathcal{S}$ answers as follows: if there exists some d such that (ϱ, d) already belongs to $\mathfrak{H}_1$, $\mathcal{S}$ returns d; else, $\mathcal{S}$ provides $\mathcal{A}$ with $d \in_R \{0,1\}^l$, and appends (ϱ, d) to $\mathfrak{H}_1$.

(3) At H digest query on some $\psi = (\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, $\mathcal{S}$ responds as follows: if (ψ, κ) already belongs to $\mathfrak{H}_2$, for some κ, $\mathcal{S}$ returns κ; else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, provides $\mathcal{A}$ with κ, and appends (ψ, κ) to $\mathfrak{H}_2$.

(4) At *Send*$(\hat{P}_l, \hat{P}_m)$ query, $\mathcal{S}$ chooses $x \in_R [1, q-1]$, creates a session with identifier $(\hat{P}_l, \hat{P}_m, X, \star, \mathcal{I})$, and provides $\mathcal{A}$ with the message $(\hat{P}_l, \hat{P}_m, X)$.

(5) At *Send*$(\hat{P}_m, \hat{P}_l, Y)$ query, $\mathcal{S}$ chooses $x \in_R [1, q-1]$, creates a session with identifier $(\hat{P}_l, \hat{P}_m, X, Y, \mathcal{R})$, provides $\mathcal{A}$ with the message $(\hat{P}_l, \hat{P}_m, X)$, and completes the session $(\hat{P}_l, \hat{P}_m, X, Y, \mathcal{R})$ ($\mathcal{S}$ also updates $\mathfrak{H}_1$ and $\mathfrak{H}_2$ in this step).

(6) At *Send*$(\hat{P}_l, \hat{P}_m, X, Y)$ query, $\mathcal{S}$ updates the identifier $(\hat{P}_l, \hat{P}_m, X, \star, \mathcal{I})$ (if any) to $sid = (\hat{P}_l, \hat{P}_m, X, Y, \mathcal{I})$. If the sid' session exists and is already completed, $\mathcal{S}$ sets the sid session key to that of sid'. Else, if a digest query was previously issued on some $\psi = (\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, and if σ is the sid session signature ($\mathcal{S}$ can compute the session signature), $\mathcal{S}$ sets the session key to $H(\psi)$. Else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, sets the session key to κ, and updates $\mathfrak{H}_2$.

(7) If $\mathcal{A}$ issues a *Corrupt*$_{SC}$, an *EphemeralKeyReveal*, a *SessionKeyReveal*, or an *EstablishParty* query at a party following the first implementation approach, $\mathcal{S}$ answers faithfully.

(8) If $\mathcal{A}$ issues a *Corrupt*$_{SC}$, a *SessionKeyReveal*, a *SecretExponentReveal*, a *SessionSignatureReveal*, or an *EstablishParty* query at a party following the second implementation approach, $\mathcal{S}$ answers faithfully.

(9) At the activation of the t–th session at $\hat{A}$, if the peer is not $\hat{B}$, $\mathcal{S}$ aborts; else, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{A}, \hat{B}, X_0)$ (recall that $\mathcal{S}$ takes as input X_0 and $Y_0 \in_R \mathcal{G}^*$).

(10) At the activation of the session matching the t–th session at $\hat{A}$, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0)$.

(11) In any of the following situations, $\mathcal{S}$ aborts.
- $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
- $\mathcal{A}$ issues a *SessionKeyReveal* or an *EphemeralKeyReveal* query on the t–th session at $\hat{A}$ or its matching session.
- $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

(12) If $\mathcal{A}$ provides a guess σ_0 of the test session signature, $\mathcal{S}$ outputs

$$\begin{aligned}\left(\sigma_0 (X_0^{d_0} A)^{-b} Y_0^{-ae_0}\right)^{(d_0 e_0)^{-1}} &= \left((X_0^{d_0} A)^{y_0 e_0} Y_0^{-ae_0}\right)^{(d_0 e_0)^{-1}} \\ &= \left((Y_0^{e_0})^{x_0 d_0 + a} Y_0^{-ae_0}\right)^{(d_0 e_0)^{-1}}\end{aligned}$$

as a guess for $CDH(X_0, Y_0)$. Otherwise $\mathcal{S}$ aborts.

The simulated environment is perfect except with negligible probability; and if $\mathcal{A}$ is polynomial, so is $\mathcal{S}$. When $\mathcal{A}$ activates the test session and its matching session, the ephemeral keys X_0 and Y_0 it is provided with are chosen uniformly at random in $\mathcal{G}^*$; their distribution is the same as that of the real X and Y. The probability of guessing correctly the test session is $(n_1^2 m)^{-1}$; and if $\mathcal{S}$ guesses correctly the test session and $\mathsf{E.1.1.1}$ occurs, $\mathcal{S}$ does not abort. Thus $\mathcal{S}$ succeeds with probability $(n_1^2 m)^{-1} \Pr(\mathsf{E.1.1.1})$ which is non–negligible, unless $\Pr(\mathsf{E.1.1.1})$ is negligible. This shows that under the CDH assumption and RO model, $\mathsf{E.1.1.1}$ cannot occur, except with negligible probability.

Event $\mathsf{E.1.1.2}$. If $\mathsf{E.1.1.2}$ occurs with non–negligible probability, using $\mathcal{A}$, we build a polynomial time CDH solver, which succeeds with non–negligible probability. For this purpose, we modify the simulation in the analysis of $\mathsf{E.1.1.1}$ as follows.

- $\mathcal{S}$ takes as input $A, B \in_R \mathcal{G}^*$.
- $\hat{A}$ and $\hat{B}$'s public keys are set to A and B; the corresponding private keys are unknown. ($\mathcal{S}$ also keeps a list of the completed session identifiers together with the session keys).
- At $Send(\hat{P}_m, \hat{P}_l, Y)$ query, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ responds as follows.
  - $\mathcal{S}$ chooses $x \in_R [1, q-1]$, computes $X = G^x$, creates a session with identifier $sid' = (\hat{P}_l, \hat{P}_m, X, Y, \mathcal{R})$, and provides $\mathcal{A}$ with the message $(\hat{P}_l, \hat{P}_m, X)$.
  - $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, $d, e \in_R \{0,1\}^l$ and sets $\bar{H}(X, Y, \hat{P}_m, \hat{P}_l) = d$, $\bar{H}(Y, X, \hat{P}_m, \hat{P}_l) = e$, and the sid' session key to κ.
- At $Send(\hat{P}_l, \hat{P}_m, X, Y)$ query, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ does the following.
  - $\mathcal{S}$ updates the session identifier $(\hat{P}_l, \hat{P}_m, X, \star, \mathcal{I})$ (if any) to $sid = (\hat{P}_l, \hat{P}_m, X, Y, \mathcal{I})$.
  - And, (i) if a value is already assigned to the sid' session key, $\mathcal{S}$ sets the sid session key to that of sid'. (ii) Else, if a digest query was previously issued on some $\psi = (\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, and if $\sigma = CDH(X^d P_l, Y^e P_m)$ (in this case, d and e are already defined, and the verification is performed using the DDHO), $\mathcal{S}$ sets the sid session key to $H(\psi)$. (iii) Else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, and sets the sid session key to κ; if no value was previously assigned to $h_1 = \bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $h_2 = \bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $h_1 = d$ (resp. $h_2 = d$).
- At $\mathcal{A}$'s digest query on $\psi = (\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, or $\hat{P}_m = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ responds as follows.
  - If there is some κ such that (ψ, κ) already belongs to $\mathfrak{H}_2$, $\mathcal{S}$ returns κ.
  - Else, (i) if there is an already completed session with identifier $sid = (\hat{P}_l, \hat{P}_m, X, Y, \mathcal{I})$ or sid', and if $\sigma = CDH(X^d P_l, Y^e P_m)$, then $\mathcal{S}$ returns the completed session's key. (ii) Else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, sets $H(\psi) = \kappa$, and provides $\mathcal{A}$ with κ; if no value was previously assigned to $h_1 = \bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $h_2 = \bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $h_1 = d$ (resp. $h_2 = d$).
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if the peer is not $\hat{B}$, $\mathcal{S}$ aborts; else $\mathcal{S}$ chooses $x_0 \in_R [1, q-1]$, and provides $\mathcal{A}$ with the message $(\hat{A}, \hat{B}, X_0 = G^{x_0})$.
- When $\mathcal{A}$ activates the session matching the t–th session at $\hat{A}$, $\mathcal{S}$ chooses $y_0, \in_R [1, q-1]$, and provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0 = G^{y_0})$.
- If $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the t–th session at $\hat{A}$ or its matching session, $\mathcal{S}$ answers faithfully.
- $\mathcal{S}$ aborts in any of the following situations:
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$;
  - $\mathcal{A}$ issues a *SessionKeyReveal* query on the t–th session at $\hat{A}$ or its matching session;
  - $\mathcal{A}$ issues a $Corrupt_{SC}$ or an *EstablishParty* query on $\hat{A}$ or $\hat{B}$;
- If $\mathcal{A}$ halts with a guess σ_0 fo the test session signature, $\mathcal{S}$ outputs a guess of $CDH(A, B)$ from σ_0, x_0, y_0, d_0, and e_0.

Under the RO model, the simulation remains perfect, except with negligible probability. And, if E.1.1.2 occurs with non–negligible probability, $\mathcal{A}$ succeeds with non–negligible probability under this simulation. If $\mathcal{A}$ succeeds and $\mathcal{S}$ guesses correctly the test session (this happens with probability $(n_1^2 m)^{-1}\Pr(\mathsf{E.1.1.2})$), $\mathcal{S}$ outputs $CDH(A, B)$. Under the GDH assumption and the RO model, E.1.1.2 cannot occur, unless with negligible probability.

Events E.1.1.3 and E.1.1.4. The roles of $\hat{A}$ and $\hat{B}$ in E.1.1.3 and E.1.1.4 are symmetrical; it then suffices to discuss E.1.1.3. If E.1.1.3 occurs with non–negligible probability, using $\mathcal{A}$, we build a polynomial time CDH solver which succeeds with non–negligible probability. We modify the simulation in the analysis if E.1.1.1 as follows.

- $\mathcal{S}$ takes as input $X_0, B \in_R \mathcal{G}^*$.
- $\hat{B}$'s public key is set to B (the corresponding private key is unknown), and $\hat{A}$'s key pair is $(a = p_i, G^a), p_i \in_R [1, q-1]$.
- At $Send(\hat{P}_m, \hat{B}, X)$ query, $\mathcal{S}$ responds as follows. (*i*) $\mathcal{S}$ chooses $y \in_R [1, q-1]$, computes $Y = G^y$, creates a session with identifier $sid' = (\hat{B}, \hat{P}_m, Y, X, \mathcal{R})$, and provides $\mathcal{A}$ with the message $(\hat{B}, \hat{P}_m, Y)$. (*ii*) $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, $d, e \in_R \{0,1\}^l$, sets the sid' session key to κ, $\bar{H}(X, Y, \hat{P}_m, \hat{B}) = d$, and $\bar{H}(Y, X, \hat{P}_m, \hat{B}) = e$.
- At $Send(\hat{B}, \hat{P}_m, Y, X)$ query:
  - $\mathcal{S}$ updates the session identifier $(\hat{B}, \hat{P}_m, Y, \star, \mathcal{I})$ (if any) to $sid = (\hat{B}, \hat{P}_m, Y, X, \mathcal{I})$.
  - And, (*i*) if a value is already assigned to the sid' session key, $\mathcal{S}$ sets the sid session key to that of sid'. (*ii*) Else, if a digest query was previously issued on some $\psi = (\sigma, \hat{B}, \hat{P}_m, Y, X)$ (in this case, d and e are defined) and if $\sigma = CDH(X^d P_m, Y^e B)$, $\mathcal{S}$ sets the sid session key to $H(\psi)$. (*iii*) Else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$ and sets the sid session key to κ; if no value was previously assigned to $h_1 = \bar{H}(Y, X, \hat{B}, \hat{P}_m)$ (resp. $h_2 = \bar{H}(X, Y, \hat{B}, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $h_1 = d$ (resp. $h_2 = d$).
- At $\mathcal{A}$'s digest query on some $\psi = (\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, with $\hat{P}_l = \hat{B}$ or $\hat{P}_m = \hat{B}$, $\mathcal{S}$ responds as follows. (*i*) If the same query was previously issued, $\mathcal{S}$ returns the previously returned value. (*ii*) Else, if there is an already completed session with identifier $sid = (\hat{P}_l, \hat{P}_m, X, Y, \mathcal{I})$ or sid', and if $\sigma = CDH(X^d P_l, Y^e P_m)$, $\mathcal{S}$ returns the completed session's key. (*iii*) Else, $\mathcal{S}$ chooses $\kappa \in_R \{0,1\}^\lambda$, sets $H(\psi) = \kappa$, and provides $\mathcal{A}$ with κ. If no value was previously assigned to $h_1 = \bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $h_2 = \bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^\lambda$ and sets $h_1 = d$ (resp. $h_2 = d$).
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if the peer is not $\hat{B}$, $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{A}, \hat{B}, X_0)$ (recall the solver takes as input X_0 and B).
- When $\mathcal{A}$ activates the session matching the t–th session at $\hat{A}$, $\mathcal{S}$ chooses $y_0 \in_R [1, q-1]$, and provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0)$.
- If $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the session matching the t–th session at $\hat{A}$, $\mathcal{S}$ answers faithfully.
- In any of the following situations, $\mathcal{S}$ aborts.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a $Corrupt_{SC}$ query on $\hat{B}$ or an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.
  - $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the t–th session at $\hat{A}$.
- If $\mathcal{A}$ halts with a guess σ_0, $\mathcal{S}$ produces $\left(\sigma_0 (X_0^{d_0} A)^{-y_0 e_0} B^{-a}\right)^{e_0^{-1}}$ as a guess for $CDH(X_0, B)$.

The simulation remains perfect, except with negligible probability; the solver $\mathcal{S}$ guesses correctly the test session with probability $(n_1^2 m)^{-1}$. If $\mathcal{A}$ succeeds under this simulation, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs $CDH(X_0, B)$. Hence if $\mathcal{A}$ succeeds with non–negligible probability in E.1.1.3, $\mathcal{S}$ outputs with non–negligible probability $CDH(X_0, B)$, contradicting the GDH assumption.

None of the events E.1.1.1, E.1.1.2, E.1.1.3 or E.1.1.4 can occur with non–negligible probability; E.1.1 cannot occur, unless with negligible probability.

Analysis of E.1.2. Suppose that E.1.2 occurs with non–negligible probability, we derive from $\mathcal{A}$ a polynomial time CDH solver which succeeds with non–negligible probability. The strongest queries that $\mathcal{S}$ can issue on $\hat{A}$, $\hat{B}$, the test session and its matching session are *Corrupt*$_{SC}$ queries on both $\hat{A}$ and $\hat{B}$. (Recall that both $\hat{A}$ and $\hat{B}$ follow the second approach). We modify the simulation in the analysis of E.1.1.1 as follows.

- $\mathcal{S}$ takes $X_0, Y_0 \in_R \mathcal{G}^*$ as input.
- $\mathcal{A}$'s environment, is simulated in the same way as in the analysis of E.1.1.1, except that i and j are chosen in $\{k \mid \hat{P}_k$ follows the second implementation approach$\}$ (we suppose here that $n - n_1 \geqslant 2$, and still refer to $\hat{P}_i$ as $\hat{A}$ and $\hat{P}_j$ as $\hat{B}$).
- $\mathcal{S}$ aborts in the following situations.
  - $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a *SessionKeyReveal*, a *SecretExponentReveal*, or a *SessionSignatureReveal* query on the test session or its matching session.

The simulation remains prefect, and if $\mathcal{A}$ is polynomial, so is $\mathcal{S}$. In addition, $\mathcal{S}$ guesses correctly the test session with probability $((n-n_1)^2 m)^{-1}$; and if $\mathcal{A}$ succeeds and $\mathcal{S}$ guesses correctly the test session, it outputs $CDH(X_0, Y_0)$ (from $\mathcal{A}$'s forgery a, b, d_0 and e_0). $\mathcal{S}$ succeeds with probability $((n-n_1)^2 m)^{-1}\,\Pr(\mathsf{E.1.2})$ which is non–negligible, unless $\Pr(\mathsf{E.1.2})$ is negligible. Under the GDH assumption and the RO model, E.1.2 cannot occur with non–negligible probability.

Analysis of E.1.3. In E.1.3 ($\hat{A}$ and $\hat{B}$ follow different implementation approaches), either $\hat{A}$ or $\hat{B}$ follows the first implementation approach; we suppose that $\hat{A}$ follows the first implementation approach. (As the test session's matching session exists, from any polynomial time machine which succeeds in E.1.3 when $\hat{A}$ follows the first approach, one can derive a polynomial time machine which succeeds with the same probability when $\hat{A}$ follows the second approach.) The strongest queries that $\mathcal{A}$ can perform on $\hat{A}$, $\hat{B}$, the test session, and its matching session are (*i*) *Corrupt*$_{SC}$ queries on both $\hat{A}$ and $\hat{B}$, (*ii*) an *EphemeralKeyReveal* query on the test session and a *Corrupt*$_{SC}$ query on $\hat{B}$. And, since from any polynomial time machine which succeeds in E.1.3, and issues weaker queries, one can build a polynomial time machine which succeeds with the same probability and performs one of the above strongest queries, it suffices to consider the following events.
E.1.3.1: "E.1.3 $\wedge$ $\mathcal{A}$ issues *Corrupt*$_{SC}$ queries on both $\hat{A}$ and $\hat{B}$";
E.1.3.2: "E.1.3 $\wedge$ $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the test session and a *Corrupt*$_{SC}$ query on $\hat{B}$."
To show that E.1.3.1 cannot occur with non–negligible probability, we use the simulation in the analysis of E.1.1.1, modified as follows.

- The environment remains the same except that $i \in_R \{k \mid \hat{P}_k$ follows the first implementation approach$\}$, and $j \in_R \{k \mid \hat{P}_k$ follows the second implementation approach$\}$.
- $\mathcal{S}$ aborts in any of the following situations.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a *SessionKeyReveal* query on the t–th session at $\hat{A}$ or its matching session.

- $\mathcal{A}$ issues a *SecretExponentReveal*, or a *SessionSignatureReveal* query on the session matching the test session, or an *EphemeralKeyReveal* query on the test session.
- $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

Using the same arguments, as in the analysis of E.1.1.1, $\mathcal{S}$ is a polynomial time CDH solver which succeeds with probability $(n_1(n-n_1)m)^{-1}\operatorname{Pr}(\mathsf{E.1.3.1})$. Under the GDH assumption and the RO model, Pr(E.1.3.1) is negligible.

Making $\mathcal{S}$ take as input $X_0, B \in_R \mathcal{G}^*$ (the arguments are similar to that used in the analysis of the event E.1.1.3), one can show also that E.1.3.2 cannot occur, unless with negligible probability.

Analysis of E.2

Suppose that E.2 ($\mathcal{A}$ succeeds in forging the signature of some fresh session without matching session) occurs with non negligible probability. As E.2 divides in
E.2.1: "E.2 $\wedge$ both $\hat{A}$ and $\hat{B}$ follow the first implementation approach";
E.2.2: "E.2 $\wedge$ both $\hat{A}$ and $\hat{B}$ follow the second implementation approach";
E.2.3: "E.2 $\wedge$ $\hat{A}$ and $\hat{B}$ follow different implementation approaches";
at least one of the events E.2.1, E.2.2, or E.2.3 occurs with non–negligible probability.

Event E.2.1. The strongest queries that $\mathcal{A}$ can perform in E.2.1 are either an *EphemeralKeyReveal* query on the test session, or a *Corrupt*$_{SC}$ query on $\hat{A}$. It then suffices to discuss E.2.1.1: "E.2.1 $\wedge\,\mathcal{A}$ performs a *Corrupt*$_{SC}$ query on $\hat{A}$," and E.2.1.2: "E.2.1 $\wedge\,\mathcal{A}$ performs an *EphemeralKeyReveal* query on the test session."
E.2.1.1. To show that E.2.1.1 cannot happen with non–negligible probability, we modify the simulation in the analysis of E.1.1.3 to take as input $a \in_R [1, q-1]$ and $X_0, B \in_R \mathcal{G}^*$ ($\hat{A}$'s key pair is set to (a, G^a), and $\hat{B}$'s public key to B); $\mathcal{S}$ aborts if $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$. The simulation remains perfect, except with negligible probability. And if $\mathcal{S}$ guesses correctly the test session, and $\mathcal{A}$ succeeds with a forgery σ_0, $\mathcal{S}$ outputs σ_0 as a FDCR–1 forgery, on messages $\hat{A}$ and $\hat{B}$ with respect to the public keys A and B. $\mathcal{S}$ succeeds with probability $((n-n_1)^2m)^{-1}\operatorname{Pr}(\mathsf{E.2.1.1})$, and contradicts Proposition 3, unless Pr(E.2.1.1) is negligible.
E.2.1.2. We modify here the simulation in the analysis of E.1.1.2 to abort if $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$. The simulated environment remains perfect, except with negligible probability. And from any valid forgery σ_0, and a correct guess of the test session, $\mathcal{S}$ outputs $A^{y_0e_0+b}$ (from σ_0, x_0, d_0, and e_0). $\mathcal{S}$ is polynomial; and if E.2.1.2 occurs with non–negligible probability, on input $A, B \in_R \mathcal{G}^*$, $\mathcal{S}$ outputs Y_0 and $A^{y_0e_0+b}$ with non–negligible probability. Hence, using the "oracle replay technique" [27], $\mathcal{S}$ yields a polynomial time CHD solver, which succeeds with non–negligible probability; contradicting the GDH assumption.

Event E.2.2. There are two cases to distinguish in the analysis of E.2.2. The reason is the difficulties that arise when simulating a session initiated at $\hat{B}$ without matching session. Suppose an attacker, which does the following at some point of its execution: (1) activate a session with initiator $\hat{B}$ and peer $\hat{P}_i$, (2) issue digest queries on $(Y, Z_i, \hat{B}, \hat{P}_i)$, for arbitrary Z_is, where Y is the ephemeral public key received, from $\hat{B}$ at session activation, and (3) issue *Send*(B, P_i, Y, Z_{i_0}) where Z_{i_0} equals some Z_i, and (4) issue *SecretExponentReveal*(B, P_i, Y, Z_{i_0}). In this case, as the digest value e has to be set before the incoming ephemeral public key is known, we cannot simulate consistently the *SecretExponentReveal* query in sessions at $\hat{B}$. 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 4 Seq1

```
(1) Issue $Send(B, P_i)$ to obtain $Y$.
(2) Issue an arbitrary number of digest queries on $(Y, Z_i, \hat{B}, \hat{P}_i)$, where $Z_i \in \mathcal{G}^*$.
(3) Choose $Z_i' \in_R \{Z_i\}$ and issue $Send(\hat{P}_i, \hat{B}, Y, Z_i')$.
(4) Issue a $SecretExponentReveal$ or a $SessionSignatureReveal$ query on the session
    $(\hat{B}, \hat{P}_i, Y, Z_i', \mathcal{I})$.
```

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).

Consider a polynomial time attacker $\mathcal{A} \notin \mathfrak{B}$, and suppose that E.2.2 occurs with non–negligible probability. Using $\mathcal{A}$, we build a polynomial time FXCR–1 signature forger, which succeeds with non–negligible probability. For this purpose, we modify the simulation in the analysis of E.1.1.1 as follows.

- $\mathcal{S}$ takes as input $X_0, B \in_R \mathcal{G}^*$.
- Both $i, j \in_R \{k \mid \hat{P}_k$ follows the second implementation approach$\}$; $\hat{A}$'s key pair is set to $(a = p_i, G^{p_i})$, $p_i \in_R [1, q-1]$ and $\hat{B}$'s public key to B; the corresponding private key is unknown (we suppose that $\hat{A} \neq \hat{B}$).
- At $Send(\hat{P}_l, \hat{B}, X)$ query, $\mathcal{S}$ answers as follows.
  - $\mathcal{S}$ chooses $s_B \in_R [1, q-1], d \in_R \{0,1\}^l$, and sets $Y = (G^{s_B} B^{-1})^{d^{-1}}$. If there is some d' such that $((X, Y, \hat{P}_l, \hat{B}), d')$ already belongs to $\mathfrak{H}_1$, $\mathcal{S}$ aborts; else, $\mathcal{S}$ appends $((X, Y, \hat{P}_l, \hat{B}), d)$ to $\mathfrak{H}_1$.
  - $\mathcal{S}$ creates a session with identifier $sid' = (\hat{B}, \hat{P}_l, Y, X, \mathcal{R})$, completes the sid' session, and provides $\mathcal{A}$ with the message $(\hat{B}, \hat{P}_l, Y)$. (Notice that $\mathcal{S}$ can compute the session signature.)
- At $\mathcal{A}$'s $Send(\hat{B}, \hat{P}_l)$ query, $\mathcal{S}$ responds as follows.
  - $\mathcal{S}$ chooses $s_B \in_R [1, q-1], e \in_R \{0,1\}^l$, and sets $Y = (G^{s_B} B^{-1})^{e^{-1}}$. If there exists some X and e' such that $((Y, X, \hat{B}, \hat{P}_l,), e')$ already belongs to $\mathfrak{H}_1$, $\mathcal{S}$ aborts.
  - $\mathcal{S}$ creates a session with identifier $(\hat{B}, \hat{P}_l, Y, \star, \mathcal{I})$, and provides $\mathcal{A}$ with $(\hat{B}, \hat{P}_l, Y)$. Later, when $\mathcal{A}$ issues $Send(\hat{B}, \hat{P}_l, Y, X)$, $\mathcal{S}$ sets $e = H_1(Y, X, \hat{B}, \hat{P}_l)$, and completes the session $(\hat{B}, \hat{P}_l, Y, X, \mathcal{I})$.
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if the peer is not $\hat{B}$, $\mathcal{S}$ aborts; else, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{A}, \hat{B}, X_0)$.
- $\mathcal{S}$ aborts in any of the following situations.
  - $\mathcal{A}$ activates at $\hat{B}$ a session matching the t–th session at $\hat{A}$.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a $Corrupt_{SC}$ query on $\hat{B}$, or an $EstablishParty$ query on $\hat{A}$ or $\hat{B}$.
  - $\mathcal{A}$ issues a $SecretExponentReveal$, a $SessionSignatureReveal$, or a $SessionKeyReveal$ query on the t–th session at $\hat{A}$.
- If $\mathcal{A}$ halts with a guess σ_0 of the test session signature, $\mathcal{S}$ outputs

$$\left(\sigma_0 (Y_0^{e_0} B)^{-a}\right)^{d_0^{-1}} = X_0^{y_0 e_0 + b}$$

as a guess for a FXCR–1 forgery on challenge X_0 and message $(\hat{A}, \hat{B})$ (the concatenation of $\hat{A}$ and $\hat{B}$) with respect to the public key B.

Under the RO model, the simulation of $\mathcal{A}$'s environment is perfect, except with negligible probability. The deviation happens when the same Y is chosen twice as outgoing ephemeral key in sessions at $\hat{B}$, with the same peer $\hat{P}_l$, this happens with probability less than m/q (which is negligible). Hence, under this simulation $\mathsf{E.2.2}$ occurs with non–negligible probability. And, when $\mathcal{A}$ outputs a correct forgery, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs a valid FXCR–1 signature forgery on challenge X_0 and message $(\hat{A}, \hat{B})$ with respect to the public key B. $\mathcal{S}$ succeeds with probability $\left((n-n_1)^2 m\right)^{-1} \Pr(\mathsf{E.2.2})$, where negligible terms are ignored, contradicting Proposition 2. Hence for attackers not in $\mathfrak{B}$, $\mathsf{E.2.2}$ cannot occur, unless with negligible probability.

For attackers in $\mathfrak{B}$, we will not provide a simulation; instead, we show that their success probability is bounded by the success probability of a class of attackers which can be efficiently simulated. Let $\mathcal{B}$ be an attacker in $\mathfrak{B}$, and let $d(|q|)$ be an upper bound on the number of Z_i the attacker chooses at step 2 of $\mathsf{Seq1}$, for simplicity (in the notations), we suppose that whenever $\mathcal{B}$ executes $\mathsf{Seq1}$, it chooses $d(|q|)$ Z_is at step 2. Recall that there are n parties, and each party is activated at most m times (n and m are polynomial in $|q|$).

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 $\mathbf{v} = \left((i_1^0, \cdots, i_m^0), (Z_{11}, \cdots, Z_{1d}), \cdots, (Z_{m1}, \cdots, Z_{md})\right)$, where $Z_{ij} \in_R \mathcal{G}^*$ and $i_i^0 \in_R [1, d]$, and performs *exactly* the same way as $\mathcal{B}$, except that whenever $\mathcal{B}$ executes the sequence of queries $\mathsf{Seq1}$ for the l–th time, $\mathcal{B}_R$ executes the modified sequence $\mathsf{Seq2}$. And, when $\mathcal{B}$ uses, for any other computation, a Z_i chosen during the l–th execution $\mathsf{Seq1}$, $\mathcal{B}_R$ uses Z_{li}.

Algorithm 5 Seq2

```
1. Issue $Send(\hat{B}, \hat{P}_i)$ to obtain $Y$.
2. Issue digest queries on $(Y, Z_{li}, \hat{B}, \hat{P}_i)$, for $Z_{li} \in \{Z_{l1}, \cdots, Z_{ld}\}$.
3. Issue $Send(\hat{B}, \hat{P}_i, Y, Z_{i_l^0})$.
4. Issue a $SecretExponentReveal$ or an $SessionSignatureReveal$ query on the session
   $(\hat{B}, \hat{P}_i, Y, Z_{i_l^0}, \mathcal{I})$.
```

Notice that if $\mathcal{B}$ is polynomial, then so is $\mathcal{B}_R$. Let $\mathbf{V}$ be the set of resource vectors, and t the number of times $\mathcal{B}$ executes $\mathsf{Seq1}$ (as each party is activated at most m times, and each execution of $\mathsf{Seq1}$ activates $\hat{B}$, $t \leqslant m$). For $\mathbf{v} \in \mathbf{V}$, we say that $\mathcal{B}_R(\mathbf{v})$ *matches* $\mathcal{B}$, if for all $l \in [1, t]$, the l–th time $\mathcal{B}$ executes $\mathsf{Seq1}$, it chooses $\{Z_{l1}, \cdots, Z_{ld}\}$ at step 2, and poses $\mathsf{Send}(\hat{B}, \hat{P}_i, Y, Zi_l^0)$ at step 3. Notice that if $\mathcal{B}_R(\mathbf{v})$ matches $\mathcal{B}$, $\Pr(\mathsf{E.2.2}_{\mathcal{B}}) = \Pr(\mathsf{E.2.2}_{\mathcal{B}_R(\mathbf{v})})$.

For $\mathcal{B} \in \mathfrak{B}$, we say $\mathbf{v} \in \mathbf{V}$ *possible* if there is nonzero 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 $\mathsf{Seq1}$), hence

$$\Pr(\mathsf{E.2.2}_{\mathcal{B}}) \leqslant \max_{\mathbf{v} \in Poss(\mathbf{V})} \Pr(\mathsf{E.2.2}_{\mathcal{B}_R(\mathbf{v})}).$$

To show that the success probability in $\mathsf{E.2.2}$ of an attacker $\mathcal{B}$ in $\mathfrak{B}$ is negligible, it suffices to show that $\Pr(\mathsf{E.2.2}_{\mathcal{B}_R(\mathbf{v})})$ is negligible for all $\mathbf{v} \in Poss(\mathbf{V})$. For this purpose, we provide the simulator with $\mathbf{v}$ (recall that we combine the simulator and the attacker $\mathcal{B}_R$ to build a FXCR–1 forger), and modify the activation of the sessions initiated at $\hat{B}$ as follows (the other parts of the simulation remain unchanged):

- When the attacker issues $\mathsf{Send}(B, P_i)$ for the l–th time, 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 $e = H_1(Y, Z_{i_l^0}, \hat{B}, \hat{P}_l)$.
  - Creates a session with identifier $(B, \hat{P}_i, Y, \star, \mathcal{I})$, and provides $\mathcal{A}$ with $(\hat{B}, \hat{P}_l, Y)$.
- When the attacker issues a *SecretExponentReveal*$(\mathcal{I}, B, P_i, Y, Z_{i_l^0})$, the simulator provides the attacker with s_B.

The simulation is consistent for all $\mathbf{v} \in \mathit{Poss}(\mathbf{V})$ and $\mathcal{B}_R$. As $\mathcal{S}$ knows, from the resource vector, what will be the incoming ephemeral public key, *SecretExponentReveal* and *SessionSignatureReveal* queries are consistently simulated. If $\mathcal{B}_R(\mathbf{v})$ succeeds in $\mathsf{E.2.2}$ with non–negligible probability, $\mathcal{S}$ succeeds in FXCR–1 forgery with non–negligible probability. Hence $\Pr(\mathsf{E.2.2}_{\mathcal{B}_R(\mathbf{v})})$ is negligible, for all $\mathbf{v} \in \mathit{Poss}(\mathbf{V})$ and $\mathcal{B}_R$. This implies $\Pr(\mathsf{E.2.2}_{\mathcal{B}})$ is negligible.

Event $\mathsf{E.2.3}$. The test session's matching session does not exist, and $\hat{A}$ and $\hat{B}$ follow different implementation approaches.

- If $\hat{A}$ follows the first implementation approach ($\mathsf{E.2.3.1}$), $\mathcal{A}$ is allowed to issue either a $\mathit{Corrupt}_{SC}$ query on $\hat{A}$, or an *EphemeralKeyReveal* query on the test session.
  - If $\mathsf{E.2.3.1.1}$: "$\mathsf{E.2.3.1} \wedge \mathcal{A}$ issues a $\mathit{Corrupt}_{SC}$ query on $\hat{A}$," occurs with non–negligible probability. We modify the simulation in the analysis of $\mathsf{E.1.1.1}$ to take as input $X_0, B \in_R \mathcal{G}^*$, and simulate $\hat{B}$'s role as in the analysis of $\mathsf{E.2.2}$ ($\hat{A}$'s role is simulated as in $\mathsf{E.1.1.1}$). The arguments bounding the success probability of attackers in $\mathfrak{B}$ (attackers for which a consistent simulation cannot be provided) remain valid. If $\mathcal{A}$ succeeds with non–negligible probability, it yields a polynomial time FXCR–1 signature forger which succeeds with non–negligible probability; contradicting Proposition 2.
  - And, if $\mathsf{E.2.3.1.2}$: "$\mathsf{E.2.3.1} \wedge \mathcal{A}$ issues an *EphemeralKeyReveal* query on the test session," occurs with non–negligible probability, we modify the simulation in $\mathsf{E.1.1.1}$ to take as input $A, B \in_R \mathcal{G}^*$, and abort if $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$. We simulate $\hat{A}$'s role as in $\mathsf{E.1.1.2}$ and $\hat{B}$'s role as in $\mathsf{E.2.2}$, reusing arguments bounding the success probability of attackers in $\mathfrak{B}$. From any valid forgery σ_0, $\mathcal{S}$ outputs $\sigma_0 (Y_0^{e_0} B)^{-x_0 d_0} = A^{y_0 e_0 + b}$; and using the oracle replay technique, $\mathcal{S}$ yields an efficient CDH solver, contradicting the GDH assumption.
- And, if $\hat{A}$ follows the second implementation approach, we make $\mathcal{S}$ take as input $A, B \in \mathcal{G}^*$, simulate $\hat{A}$'s role in the same way as that of $\hat{B}$ in $\mathsf{E.2.2}$, and $\hat{B}$'s role as in $\mathsf{E.1.1.2}$, except that when $\mathcal{A}$ activates the t–th session at $\hat{A}$, $\mathcal{S}$ chooses $x_0 \in_R [1, q-1]$ and provides $\mathcal{A}$ with $(\hat{A}, \hat{B}, X_0)$ ($\mathcal{S}$ also aborts if $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$). If $\mathcal{A}$ succeeds with non–negligible probability, $\mathcal{S}$ outputs with non–negligible probability $A^{y_0 e_0 + b}$, and using the oracle replay technique, $\mathcal{S}$ yields an efficient CDH solver; $\mathsf{E.2.3}$ cannot occur, except with negligible probability.

Reflection Attacks If $\hat{A} = \hat{B}$, $\mathsf{E.1}$ reduces to $\mathsf{E.1.1}$ and $\mathsf{E.1.2}$; in addition $\mathsf{E.1.1}$ reduces to $\mathsf{E.1.1.1}$ and $\mathsf{E.1.1.2}$. The analyses of the events $\mathsf{E.1.1.1}$, $\mathsf{E.1.1.2}$, and $\mathsf{E.1.2}$ hold if $\hat{A} = \hat{A}$; reflections attacks cannot succeed in $\mathsf{E.1}$.

In $\mathsf{E.2}$ (which reduces here to $\mathsf{E.2.1}$ and $\mathsf{E.2.2}$), $\mathsf{E.2.1}$ reduces to $\mathsf{E.2.1.2}$ (the $\mathit{Corrupt}_{SC}$ query is not allowed on $\hat{A}$), if $\mathcal{A}$ succeeds with non–negligible probability, it yields a polynomial time machine $\mathcal{S}$ which on input A outputs with non–negligible probability Y_0 and $(Y_0^{e_0} A)^a$, and $\mathcal{S}$ yields a squaring CDH solver, contradicting the GDH assumption.

Neither $\mathsf{E.1}$ nor $\mathsf{E.2}$ can occur with non–negligible probability, the SMQV protocol is seCK–secure.