Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS

(extended version)

Berkant Ustaoglu

June 22, 2009

Abstract

LaMacchia, Lauter and Mityagin recently presented a strong security definition for authenticated key agreement strengthening the well-known Canetti-Krawczyk definition. They also described a protocol, called NAXOS, that enjoys a simple security proof in the new model. Compared to MQV and HMQV, NAXOS is less efficient and cannot be readily modified to obtain a one-pass protocol. On the other hand MQV does not have a security proof, and the HMQV security proof is extremely complicated.

This paper proposes a new authenticated key agreement protocol, called CMQV ('Combined' MQV), which incorporates design principles from MQV, HMQV and NAXOS. The new protocol achieves the efficiency of HMQV and admits a natural one-pass variant. Moreover, we present a relatively simple and intuitive proof that CMQV is secure in the LaMacchia-Lauter-Mityagin model.

Contents

1 Introduction

Researchers from IBM and Microsoft have recently proposed two-pass Diffie-Hellman authenticated key agreement protocols called HMQV [9], KEA+ [13] and NAXOS [12]. In these protocols the two communicating parties exchange static (long-term) and ephemeral (short-term) public keys, and thereafter combine them to obtain a session key. The papers [9, 13, 12] highlight certain security issues with previous related key agreement protocols and propose solutions to address those issues. The goal of this paper is to devise a new protocol that has the best of all worlds incorporated in its design.

Security models and definitions. Choo, Boyd and Hitchcock [7] compared the most commonly used security models for key agreement [4, 3, 6]. Their conclusion was that none of the models as defined provides a significant advantage over the rest of the models. Furthermore, these models fail to capture some desirable properties of key agreement. Most significantly, the adversary is not allowed to obtain certain secret information about the session that is being attacked. Krawczyk [9] addressed many of these concerns by providing a stronger version of the Canetti-Krawczyk model [6] that captures additional security properties. These desirable properties include resistance to key-compromise impersonation (KCI) attacks, weak perfect forward secrecy (wPFS), and resilience to the leakage of ephemeral private keys (LEP). More recently LaMacchia, Lauter and Mityagin [12] provided a single definition that *simultaneously* captures all these security properties. Their security model will henceforth be called the extended Canetti-Krawczyk (eCK) model.

Protocols. NAXOS is proven secure in the eCK model, but is less efficient in that it requires 4 exponentiations per party compared to 2.5 exponentiations for MQV and HMQV. In addition there is no natural modification of NAXOS to a one-pass protocol. Unlike MQV [14], the HMQV [9] protocol has a formal security proof[1]. However the proof is extremely long and complicated, and some significant (but fixable) flaws [16, 17] have been discovered. The security proof for KEA+ [13] is in a model that is weaker than eCK; for example the adversary is not allowed to obtain the static private keys of both communicating parties. Table 1 compares MQV, HMQV, KEA+ and NAXOS in terms of efficiency (number of exponentiations per party), security and underlying assumptions. (See Section 3.3 for a more detailed analysis of the efficiency of CMQV.) As usual CK stands for Canetti-Krawczyk [6], GDH refers to the Gap Diffie-Hellman assumption [19], RO is short for the random oracle model, and KEA1 is the knowledge of exponent assumption [2].

Protocol	Efficiency	Security	Assumptions
MQV	2.5	unproven	?
HMQV	2.5	CK, wPFS, KCI, LEP	KEA1, GDH, RO
KEA+	3	CK, KCI	GDH, RO
NAXOS	4	eCK	GDH, RO
CMQV	3	eCK	GDH, RO

Table 1: Protocol comparison

[1] The security proof for MQV presented in [11] is in a very restricted security model.

Goals. This paper presents the two-pass CMQV protocol that achieves the following objectives: (i) intuitive design principles; (ii) efficiency of MQV and HMQV; (iii) relatively straightforward security proof with minimal assumptions in the eCK model; and (iv) a natural one-pass variant. The security proof was inspired by the HMQV argument [9], however NAXOS' idea of hashing ephemeral private keys with static private keys is essential to show security in the eCK model. Moreover, unlike the HMQV proof, the CMQV security proof does not need the KEA1 assumption in order to demonstrate resilience to the leakage of ephemeral private keys. On the negative side, the security of CMQV is *not* tight. As in the case of HMQV, the reduction uses the Forking Lemma of Pointcheval and Stern [20, 21], which results in a highly non-tight reduction.

Organization. Section 2 outlines the extended Canetti-Krawczyk security model and formalizes the security definition. The two-pass CMQV protocol is described in Section 3, and the complete security proof is provided in Section 4. Finally, the one-pass variant of CMQV with its security argument is presented in Section 5.

Notation. Let q be a prime, and let $\mathbb{Z}_q$ denote the integers modulo q. We denote by $\mathcal{G} = \langle G \rangle$ a multiplicatively-written cyclic group of order q generated by g, and by $\mathcal{G}^*$ the set of non-identity elements in $\mathcal{G}$. For group elements $A, B, \ldots$ the corresponding lowercase letters will denote the discrete logarithms in base g; for example $a = \log_g A$, where $a \in \mathbb{Z}_q$. Key agreement protocols take place between two parties, from among a set of n parties, denoted by $\hat{A}$, $\hat{B}$ and so on. Party $\hat{A}$'s static public key is $A \in \mathcal{G}$ and its corresponding static private key is $a = \log_g A$. In general, lower case letters represent secret information, whereas upper case letters are publicly known values. Finally, the symbol "$\in_R$" means "selected uniformly at random".

Acknowledgments

This paper owes much to the suggestions, help and advice of Alfred Menezes. I also thank Hugo Krawczyk and the two anonymous referees for their valuable comments.

2 Extended Canetti-Krawzcyk security model

In this section we outline the eCK model; for further details the reader is referred to [12, 6]. In the eCK model there are n parties each modeled by a probabilistic Turing machine. Each party has a static public-private key pair together with a certificate that binds the public key to that party. We do not assume that the certifying authority (CA) requires parties to prove possession of their static private keys, but we insist that the CA verifies that the static public key of a party belongs to $\mathcal{G}^*$. For simplicity, we will only describe the model for two-pass Diffie-Hellman protocols that exchange ephemeral and static public keys – this is without loss of generality as all the protocols in Table 1 are of this kind. More precisely, two parties $\hat{A}$, $\hat{B}$ exchange static public keys $A, B \in \mathcal{G}^*$ and ephemeral public keys $X, Y \in \mathcal{G}^*$; the session key is obtained by combining A, B, X, Y and possibly the identities $\hat{A}$, $\hat{B}$.

Sessions. A party $\hat{A}$ can be activated to execute an instance of the protocol called a *session*. Activation is made via an incoming message that has one of the following forms: (i) $(\hat{A}, \hat{B})$ or (ii) $(\hat{A}, \hat{B}, Y)$. If $\hat{A}$ was activated with $(\hat{A}, \hat{B})$ then $\hat{A}$ is the session *initiator*, otherwise the session *responder*. If $\hat{A}$ is the responder of a session then $\hat{A}$ prepares an ephemeral public key X and creates a separate session *state* where all session-specific ephemeral information is stored. The

session is identified via a session *identifier* $(\mathcal{R}, \hat{A}, \hat{B}, Y, X)$, where $\mathcal{R}$ denotes responder. If $\hat{A}$ is the initiator of a session, then $\hat{A}$ prepares an ephemeral public key X and creates a session state as in the responder case. At the onset of the protocol the initiator does not know the incoming ephemeral public key. However the session can be uniquely[2] identified with $(\mathcal{I}, \hat{A}, \hat{B}, X, \times)$, where $\mathcal{I}$ denotes initiator, and hence this string can be used as the (temporary and incomplete) session identifier. When $\hat{A}$ receives the corresponding ephemeral public key Y, the session identifier is updated to $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$. A session $(\mathcal{R}, \hat{B}, \hat{A}, X, Y)$ (if it exists) is said to be *matching* to the session $(\mathcal{I}, \hat{A}, \hat{B}, X, \times)$; it remains matching even when the identifier is updated to $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$. On the other hand, the session matching to $(\mathcal{R}, \hat{B}, \hat{A}, X, Y)$ can be any session identified by $(\mathcal{I}, \hat{A}, \hat{B}, X, \times)$ or $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$. Since it is not possible (except with negligible probability) to simultaneously have two different sessions with identifiers $(\mathcal{I}, \hat{A}, \hat{B}, X, \times)$ and $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$, a session $(\mathcal{R}, \hat{B}, \hat{A}, X, Y)$ can have at most one matching session. For a session $(*, \hat{A}, \hat{B}, *, *)$, we call $\hat{A}$ the *owner* of the session, and $\hat{B}$ the *peer* of the session.

Adversary. The adversary $\mathcal{M}$ is modeled as a probabilistic Turing machine and controls *all* communications. Parties submit outgoing messages to the adversary, who makes decisions about their delivery. The adversary presents parties with incoming messages via *Send*(message), thereby controlling the activation of sessions. The adversary does not have immediate access to a party's private information, however in order to capture possible leakage of private information the adversary is allowed the following queries:

- *EphemeralKeyReveal*(s) – The adversary obtains the ephemeral private key held by the session s.
- *SessionKeyReveal*(s) – The adversary obtains the session key for a session s, provided that the session holds a session key.
- *StaticKeyReveal*(party) – The adversary learns the static private key of the party.
- *Establish*(party) – This query allows the adversary to register a static public key on behalf of a party. In this way the adversary totally controls that party. Parties against whom the adversary did not issue this query are called *honest*.

Adversary goal. The aim of the adversary $\mathcal{M}$ is to distinguish a session key from a random key. Formally, the adversary is allowed to make one special query *Test*(s). The adversary is then given with equal probability either the session key held by s or a random key. The adversary wins the game if he guesses correctly whether the key is random or not. To define secure protocols we need the following.

Definition 2.1 (fresh session) *Let* s *be the session identifier of a completed session, owned by an honest party $\hat{A}$ with peer $\hat{B}$, who is also honest. Let* s^* *be the session identifier of the matching session of* s*, if it exists. Define* s *to be* fresh *if none of the following conditions hold:*

(i) $\mathcal{M}$ issues a SessionKeyReveal(s) query or a SessionKeyReveal(s^*) query (if s^* exists);

(ii) s^* *exists and $\mathcal{M}$ makes either of the following queries:*

- *both StaticKeyReveal($\hat{A}$) and EphemeralKeyReveal(s), or*

[2]Since ephemeral keys are selected at random on a per-session basis, the probability that an ephemeral public key X is chosen twice by $\hat{A}$ is negligible.

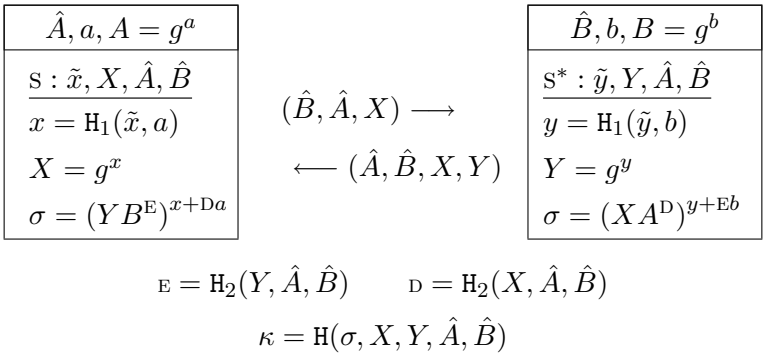

Figure 1: Two-pass CMQV

- *both StaticKeyReveal($\hat{B}$) and EphemeralKeyReveal(S^*);*

(iii) S^* *does not exist and* $\mathcal{M}$ *makes either of the following queries:*

- *both StaticKeyReveal($\hat{A}$) and EphemeralKeyReveal(*S*), or*
- *StaticKeyReveal($\hat{B}$).*

We are now ready to present the eCK security notion.

Definition 2.2 (eCK security) *A key agreement protocol is* secure *if the following conditions hold:*

1. *If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key (or both output indication of protocol failure).*
2. *No polynomially bounded adversary* $\mathcal{M}$ *can distinguish the session key of a fresh session from a randomly chosen session key, with probability greater than 1/2 plus a negligible fraction.*

The adversary $\mathcal{M}$ is allowed to continue interacting with the parties even after issuing the *Test* query. However, the test session must remain fresh through the experiment.

As mentioned at the end of Section 1, this security definition is very strong in the sense that it simultaneously captures most of the desirable security properties for authenticated key agreement that have been identified in the literature including resistance to key-compromise impersonation attacks, weak perfect forward secrecy, and resilience to the leakage of ephemeral private keys. Unlike in the CK model [6], the adversary in the eCK model is not equipped with a *SessionStateReveal* query which enables it to learn the entire session state of a particular session. This does not represent a deficiency in the eCK model since protocols such as HMQV [9] proven secure in the CK model typically specify that the ephemeral private key is the only private information stored in the session state in which case the *EphemeralKeyReveal* query is functionally equivalent to the *SessionStateReveal* query. In general, by specifying that the session specific private information (the session state) is part of the ephemeral private key, the *SessionStateReveal* and *EphemeralKeyReveal* queries can be made functionally equivalent.

3 Two-pass CMQV

Two-pass CMQV is a Diffie-Hellman authenticated key agreement protocol that aims to establish a secure session key between two parties; see Figure 1 for an informal description. In addition

to the notation adopted at the end of Section 1, let $\mathtt{H}_1 : \{0,1\}^\lambda \times \mathbb{Z}_q{}^* \to \mathbb{Z}_q{}^*$, $\mathtt{H}_2 : \{0,1\}^* \to \mathbb{Z}_q{}^*$, and $\mathtt{H} : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions modeled as random oracles.

3.1 Protocol description

We assume for the remainder of the paper that a party never executes the protocol with itself. The two-pass CMQV protocol is formally given in the following.

Definition 3.1 (two-pass CMQV protocol) *The protocol proceeds as follows:*

1. *Upon activation $(\hat{A}, \hat{B})$, party $\hat{A}$ (the initiator) performs the steps:*
   (a) *Select an ephemeral private key $x \in_R \{0,1\}^\lambda$.*
   (b) *Compute the ephemeral public key $X = g^{\mathtt{H}_1(x,a)}$.*
   (c) *Initiate session $\mathrm{S} = (\mathcal{I}, \hat{A}, \hat{B}, X, *)$ and send $(\hat{B}, \hat{A}, X)$ to $\hat{B}$.*
2. *Upon activation $(\hat{B}, \hat{A}, X)$, party $\hat{B}$ (the responder) performs the steps:*
   (a) *Verify that $X \in \mathcal{G}^*$.*
   (b) *Select an ephemeral private key $y \in_R \{0,1\}^\lambda$.*
   (c) *Compute the ephemeral public key $Y = g^{\mathtt{H}_1(y,b)}$.*
   (d) *Compute $\mathrm{E} = \mathtt{H}_2(Y, \hat{A}, \hat{B})$ and $\mathrm{D} = \mathtt{H}_2(X, \hat{A}, \hat{B})$.*
   (e) *Compute $\sigma = (XA^{\mathrm{D}})^{\mathtt{H}_1(y,b)+\mathrm{E}b}$ and $\kappa = \mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$.*
   (f) *Destroy y and σ.*
   (g) *Complete session $\mathrm{S} = (\mathcal{R}, \hat{B}, \hat{A}, X, Y)$ with session key κ and send $(\hat{A}, \hat{B}, X, Y)$ to $\hat{A}$.*
3. *Upon activation $(\hat{A}, \hat{B}, X, Y)$, party $\hat{A}$ performs the steps:*
   (a) *Verify that a session with identifier $(\mathcal{I}, \hat{A}, \hat{B}, X, \times)$ exists.*
   (b) *Verify that $Y \in \mathcal{G}^*$.*
   (c) *Compute $\mathrm{E} = \mathtt{H}_2(Y, \hat{A}, \hat{B})$ and $\mathrm{D} = \mathtt{H}_2(X, \hat{A}, \hat{B})$.*
   (d) *Compute $\sigma = (YB^{\mathrm{E}})^{\mathtt{H}_1(x,a)+\mathrm{D}a}$ and $\kappa = \mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$.*
   (e) *Destroy x and σ.*
   (f) *Complete session $\mathrm{S} = (\mathcal{I}, \hat{A}, \hat{B}, X, Y)$ with session key κ.*

If any verification fails the party erases all session specific information, which includes the ephemeral private key, from its memory and aborts the session.

It is straightforward to verify that both parties compute the same shared secret σ, and therefore also the same session key.

3.2 Design rationale

Public-key validation. Public-key validation (i.e., checking that static and ephemeral public keys belong to $\mathcal{G}^*$) prevents potential invalid-curve [1] and small subgroup attacks [15] (see also [17]). In other words, with validation a party obtains some assurance that computations involving its static private key do not reveal any information about the key itself, as long as the underlying group is cryptographically strong.

Hashing ephemeral and static private keys. A careful reader observes that in Definition 3.1 the value $x = \mathtt{H}_1(\tilde{x}, a)$ is *never* stored. Whenever $\mathtt{H}_1(\tilde{x}, a)$ is needed, it is computed. This implies that the session state does not store x. The idea is that without knowing *both* the ephemeral private key $\tilde{x}$ *and* the static private key a, no entity is able to compute the discrete logarithm x of an ephemeral public key X. This elegant idea, first described in [12], allows the protocol to attain resistance to ephemeral private key leakage without resorting to non-trivial assumptions like KEA1 [2] (as needed for HMQV [9]).

Rationale for exponents. Given a Computational Diffie-Hellman challenge with inputs $U, V \in_R \mathcal{G}$, knowledge of either of the discrete logarithms of U or V is enough to solve the CDH instance. If an adversary $\mathcal{M}$, given a static public key B, is able to find a group element Y such that $\mathcal{M}$ knows the discrete logarithm of $T = XB^{\mathrm{E}}$, then it is easy to see that $\mathcal{M}$ can impersonate $\hat{B}$ to other parties (since $\mathcal{M}$ can compute the shared secret $\sigma = (XA^{\mathrm{D}})^t$ where $t = \log_g T$, thereby impersonating $\hat{B}$ to $\hat{A}$). Defining E to depend on Y ensures that the adversary is not able to compute the discrete logarithm of YB^{E}. Moreover, including the identity of the intended peer in the derivation of E prevents the adversary from potentially benefiting from the replay of Y to two distinct parties $\hat{A}$ and $\hat{C}$. One may argue that the inclusion of $\hat{B}$'s identity in the derivation of E is not needed since σ in any case depends on $\hat{B}$'s static public key B. However, since the CA does not require parties to prove possession of their static private keys, $\mathcal{M}$ may establish a new party with static public key B. Hence $\hat{B}$ is included in the derivation of E.

We note that a very similar definition of E was used in HMQV [9]. For both HMQV and CMQV, this definition of the exponents is crucial for the security proof, but in both cases the reduction is non-tight. It is worth investigating if the requirements on E and D can be modified to attain a tight security reduction.

Session key derivation. The session key is $\kappa = \mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$. The secrecy of σ guarantees that only the intended parties can possibly compute κ. Including identities in the key derivation is a generic way to prevent unknown-key share attacks (see [5]). Furthermore, inclusion of X and Y in the key derivation allows for a simple argument that non-matching sessions have different session keys.

3.3 Efficiency of CMQV

The efficiency comparison in Table 1 is simplified; in particular, it does not take into account validation and various speedups that may be applicable. Consider the following groups of practical interest: (i) DSA-type groups (order-q subgroups of the multiplicative group of prime fields $\mathbb{F}_p$); and (ii) elliptic curves of prime order q or nearly prime order hq. Validation for DSA-type groups requires a full exponentiation; in contrast validating points on elliptic curves of prime order is essentially free. For nearly prime order curves, rather than verifying that the order of a public key is q, parties could use the corresponding public keys multiplied by the cofactor h. If the two public keys Y and B are validated, then computing $(YB^{\mathrm{E}})^{x+\mathrm{D}a}$ is equivalent to computing $Y^{s_1}B^{s_2}$, where $s_1 = x + \mathrm{D}a \bmod q$ and $s_2 = \mathrm{E}(x + \mathrm{D}a) \bmod q$. Therefore, computations can be speedup using Shamir's trick ([18, Algorithm 14.88]), reducing the cost by 0.83 exponentiations on average.

Table 2 compares CMQV with HMQV as described in [10], accounting for the validation and Shamir's speedup. The numbers in parentheses for MQV and CMQV represent the naive count of group exponentiations without accounting for possible improvements in the computations.

	DSA groups	Elliptic curves of prime order	Elliptic curves of nearly prime order
CMQV	3.17 (4)	2.17 (3)	2.17 (3)
MQV	3.17 (3.5)	2.17 (2.5)	2.17 (2.5)
HMQV-P1363	2.5 / 3.5	2.17	2.17

Table 2: Efficiency comparison in terms of group exponentiations

The numbers for HMQV correspond to the two versions of HMQV as described in [10]. For HMQV, the difference is significant only in DSA-type groups as the more efficient version avoids full validation. However, the security proof in the case where validation is not required assumes that no ephemeral private keys are leaked to the adversary.

4 Two-pass CMQV security

This section presents a formal security argument for two-pass CMQV. The *GDH assumption* in $\mathcal{G}$ is that the CDH problem in $\mathcal{G}$ cannot be solved in polynomial time with non-negligible success probability even when a DDH oracle for $\mathcal{G}$ is available.

Theorem 4.1 *If* $\mathtt{H}_1$,$\mathtt{H}_2$ *and* $\mathtt{H}$ *are random oracles, and* $\mathcal{G}$ *is a group where the GDH assumption holds, then CMQV is eCK secure.*

Proof: Verifying condition 1 of Definition 2.2 is straightforward; it remains to verify condition 2.

Let λ denote the security parameter, whence $q = |\mathcal{G}| = \Theta(2^\lambda)$. Let $\mathcal{M}$ be a polynomially (in λ) bounded CMQV adversary. The adversary $\mathcal{M}$ is said to be successful (event M) with non-negligible probability if $\mathcal{M}$ wins the distinguishing game described in Section 2 with probability $\frac{1}{2}+p(\lambda)$, where $p(\lambda)$ is non-negligible. Assume that $\mathcal{M}$ operates in an environment that involves at most $n(\lambda)$ parties, $\mathcal{M}$ activates at most $s(\lambda)$ sessions within a party, and makes at most $h_1(\lambda), h_2(\lambda)$ and $h(\lambda)$ queries to oracles $\mathtt{H}_1$,$\mathtt{H}_2$ and $\mathtt{H}$, respectively; and terminates after time at most $\mathcal{T}_\mathcal{M}$. Let the test session be $\mathrm{s}^t = (\hat{A}, \hat{B}, X, Y)$ and let H denote the event that $\mathcal{M}$ queries $\mathtt{H}$ with $(\sigma, X, Y, \hat{A}, \hat{B})$, where $\sigma = \mathrm{CDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}})$. Let $\overline{H}$ be the complement of H and s^* be any other completed session owned by an honest party, such that s^t and s^* are non-matching. Since s^t and s^* are non-matching, the input to the key derivation function $\mathtt{H}$ are different for s^t and s^*. And since $\mathtt{H}$ is a random oracle it follows that $\mathcal{M}$ cannot obtain any information about the test session key from the session key of non-matching sessions. Hence $\Pr(M \wedge \overline{H}) \leq \frac{1}{2}$ and

$$\Pr(M) = \Pr(M \wedge \overline{H}) + \Pr(M \wedge H) \leq \frac{1}{2} + \Pr(M \wedge H),$$

whence $\Pr(M \wedge H) \geq p$; henceforth the event $M \wedge H$ is denoted by M^*.

Following the standard approach such an adversary $\mathcal{M}$ is used to construct a GDH solver $\mathcal{S}$ that succeeds with non-negligible probability. Let $\xi : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ be a random function known only to $\mathcal{S}$, such that $\xi(X, Y) = \xi(Y, X)$. The algorithm $\mathcal{S}$ will use ξ to simulate $\mathrm{CDH}(X, Y)$ when $\mathcal{S}$ may not know $\log_g(X)$ or $\log_g(Y)$. Let the input to the GDH challenge be (U, V) and consider the following complementary events:

DL. There exists an honest party $\hat{B}$ such that $\mathcal{M}$, during its execution, queries $\mathtt{H}_1$ with $(*, b)$, before issuing a *StaticKeyReveal*$(\hat{B})$ query. (Note that $\mathcal{M}$ does not necessarily make a *StaticKeyReveal*$(\hat{B})$ query.)

$\overline{DL}$. During its execution, for every honest party $\hat{B}$ for which $\mathcal{M}$ queries $\mathtt{H}_1$ with $(*, b)$, $\mathcal{M}$ issued *StaticKeyReveal*$(\hat{B})$ before the first $(*, b)$ query to $\mathtt{H}_1$.

If $\mathcal{M}$ succeeds with non-negligible probability, and hence $\Pr(M^*) \geq p$, it must be the case that either event $DL \wedge M^*$ or event $\overline{DL} \wedge M^*$ occurs with non-negligible probability. These events are considered separately.

4.1 Event DL

Simulation. Suppose that event $DL \wedge M^*$ occurs with non-negligible probability. In this case $\mathcal{S}$ prepares n parties. One party, called $\hat{V}$, is selected at random and assigned static public key V; $\mathcal{S}$ represents $\hat{V}$'s static private key by $\nu \in_R \mathbb{Z}_q$. The remaining $n-1$ parties are assigned random static key pairs. The adversary $\mathcal{M}$ is initiated on this set of parties and the simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*$(\hat{A}, \hat{B})$: $\mathcal{S}$ executes Step 1 of the protocol.
2. *Send*$(\hat{B}, \hat{A}, X)$: $\mathcal{S}$ executes Step 2 of the protocol. Except if $\hat{B} = \hat{V}$, in which case $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.
3. *Send*$(\hat{A}, \hat{B}, X, Y)$: $\mathcal{S}$ executes Step 3 of the protocol. Except if $\hat{A} = \hat{V}$, in which case $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.
4. *EphemeralKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.
5. *SessionKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.
6. *StaticKeyReveal*$(\hat{A})$: $\mathcal{S}$ responds to the query faithfully, unless $\hat{A} = \hat{V}$ in which case $\mathcal{S}$ aborts with failure.
7. *Establish*$(\hat{M})$: $\mathcal{S}$ responds to the query faithfully.
8. $\mathtt{H}_1(s, c)$: $\mathcal{S}$ checks if the equation $g^c = V$ holds, in which case $\mathcal{S}$ stops $\mathcal{M}$ and is successful by outputting $\mathrm{CDH}(U, V) = U^c$; otherwise $\mathcal{S}$ simulates a random oracle in the usual way.
9. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.
10. $\mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$:
    (a) If $\hat{V} \in \{\hat{A}, \hat{B}\}$ and $\sigma \neq \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$.
        i. If $\tau = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, YB^{\mathrm{E}}), X, Y, \hat{A}, \hat{B}\right)$.
        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.
    (b) $\mathcal{S}$ simulates a random oracle in the usual way.
11. *Test*(s): $\mathcal{S}$ responds to the query faithfully.
12. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $DL \wedge M^*$. $\mathcal{S}$'s simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. With probability at least $\frac{1}{n}$, $\mathcal{S}$ assigns the public key V to an honest party $\hat{B}$ for whom $\mathcal{M}$ will query $\mathtt{H}_1(*, b)$ without first issuing a *StaticKeyReveal*($\hat{B}$) query. In this case $\mathcal{S}$ is successful as described in Step 8 and the abortions in Steps 6 and 12 do not occur. Hence if event $DL \wedge M^*$ occurs with probability p_{DL}, then $\mathcal{S}$ is successful with probability $\Pr(S)$ that is bounded by

$$\Pr(\mathcal{S}) \geq \frac{1}{n} p_{DL}. \tag{1}$$

4.2 Event $\overline{DL}$

Let TM be the event "the test session has a matching session owned by an honest party". Event $\overline{DL} \wedge M^*$ is further subdivided into the following complementary events:

(i) $T_m = (\overline{DL} \wedge M^* \wedge TM)$, and

(ii) $\overline{T_m} = (\overline{DL} \wedge M^* \wedge \overline{TM})$.

Let $p_{\tilde{DL}} = \Pr(\overline{DL} \wedge M^*)$, $p_m = \Pr(T_m)$, and $p_{\tilde{m}} = \Pr(\overline{T_m})$. Since T_m and $\overline{T_m}$ are complementary $p_{\tilde{DL}} = p_m + p_{\tilde{m}}$. Therefore, if event $\overline{DL} \wedge M^*$ occurs with non-negligible probability, then either T_m or $\overline{T_m}$ occurs with non-negligible probability. Events T_m and $\overline{T_m}$ are next considered separately.

4.2.1 Event T_m

Simulation. Suppose that event T_m occurs with non-negligible probability. In this case $\mathcal{S}$ establishes n honest parties that are assigned random static key pairs, and randomly selects two integers $i, j \in_R [1, \ldots, ns]$. The i'th and the j'th sessions created will be called s^U and s^V, respectively. The ephemeral private key of s^U is denoted by $\tilde{u}$ and the ephemeral private keys of s^V is denoted by $\tilde{v}$. The simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*($\hat{A}, \hat{B}$): $\mathcal{S}$ executes Step 1 of the protocol. However, if the session being created is s^U or s^V, then $\mathcal{S}$ deviates by setting the ephemeral public key X to be U or V, respectively, thereby defining $\mathtt{H}_1(\tilde{u}, a) = \log_g U$ or $\mathtt{H}_1(\tilde{v}, a) = \log_g V$. Note that in this case $\mathcal{S}$ cannot respond to either $\mathtt{H}_1(\tilde{u}, a) = \log_g U$ or $\mathtt{H}_1(\tilde{v}, a) = \log_g V$.

2. *Send*($\hat{B}, \hat{A}, X$): $\mathcal{S}$ executes Step 2 of the protocol. However, if the session being created is s^U or s^V, then $\mathcal{S}$ deviates by setting the ephemeral public key Y to be U or V, respectively, and setting $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.

3. *Send*($\hat{A}, \hat{B}, X, Y$): $\mathcal{S}$ executes Step 3 of the protocol. However, if $X \in \{U, V\}$ then $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.

4. *EphemeralKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

5. *SessionKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

6. *StaticKeyReveal*($\hat{A}$): $\mathcal{S}$ responds to the query faithfully.

7. *Establish*($\hat{M}$): $\mathcal{S}$ responds to the query faithfully.

8. $\mathtt{H}_1(x, a)$: $\mathcal{S}$ simulates a random oracle in the usual way except if $\hat{A}$ owns s^U and $x = \tilde{u}$ or if $\hat{A}$ owns s^V and $x = \tilde{v}$, in which case $\mathcal{S}$ aborts with failure.

9. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.

10. $\mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$:

    (a) If $\{X, Y\} = \{U, V\}$ and $\mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma) = 1$, then $\mathcal{S}$ aborts $\mathcal{M}$ and is successful by outputting $\mathrm{CDH}(U, V) = \sigma g^{-ab^{\mathrm{ED}}} X^{-b^{\mathrm{E}}} Y^{-a^{\mathrm{D}}}$.

    (b) If $X \in \{U, V\}$ and $\sigma \neq \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$.

        i. If $\tau = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, YB^{\mathrm{E}}), X, Y, \hat{A}, \hat{B}\right)$.

        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

    (c) If $Y \in \{U, V\}$ and $\sigma \neq \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$.

        i. If $\tau = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, YB^{\mathrm{E}}), X, Y, \hat{A}, \hat{B}\right)$.

        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

    (d) $\mathcal{S}$ simulates a random oracle in the usual way.

11. *Test*(s^t): If s^U and s^V are non-matching or if s^t is neither s^U nor s^V, then $\mathcal{S}$ aborts; otherwise responds to the query faithfully.

12. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $T_m \wedge \overline{DL} \wedge M^*$. $\mathcal{S}$'s simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. The probability that $\mathcal{M}$ selects s^U and s^V as the test session and its matching is at least $\frac{2}{(ns)^2}$. Suppose that this is the case, so $\mathcal{S}$ does not abort as in Step 11, and suppose that event T_m occurs. Without loss of generality, let $\mathrm{s}^t = \mathrm{s}^U = (\hat{A}, \hat{B}, U, V)$. Since $\tilde{u}$ is used only in the test session, $\mathcal{M}$ must obtain it via an *EphemeralKeyReveal* query before making an $\mathtt{H}_1$ query that includes $\tilde{u}$. Similarly, $\mathcal{M}$ must obtain $\tilde{v}$ from the matching session via an *EphemeralKeyReveal* query before making an $\mathtt{H}_1$ query that includes $\tilde{v}$. Under event $\overline{DL}$, the adversary first issues a *StaticKeyReveal* query to a party before making an $\mathtt{H}_1$ query that includes that party's static private key. Since the test session is fresh, $\mathcal{M}$ can query for at most one value in each of the pairs $(\tilde{u}, a)$ and $(\tilde{v}, b)$; hence $\mathcal{S}$ does not abort as described in Step 8. Under event M^*, except with negligible probability of guessing $\xi(UA^{\mathrm{D}}, VB^{\mathrm{E}})$, $\mathcal{S}$ is successful as described in Step 10a and does not abort as in Step 12. Therefore if event T_m occurs, then the success probability of $\mathcal{S}$ is bounded by

$$\Pr(\mathcal{S}) \geq \frac{2}{(ns)^2} p_m. \tag{2}$$

4.2.2 Event $\overline{T_m}$

Simulation. Suppose that event $\overline{T_m}$ occurs with non-negligible probability. Recall that event $\overline{T_m}$ implies that no honest party owns a session matching to the test session. In this case $\mathcal{S}$ prepares n parties. One party, called $\hat{V}$, is selected at random and assigned static public key V and $\mathcal{S}$ represents $\hat{V}$'s static private key by $\nu \in_R \mathbb{Z}_q$. The remaining $n - 1$ parties are assigned random static key pairs. Furthermore, $\mathcal{S}$ randomly selects an integer $i \in_R [1, \ldots, ns]$. The i'th session created will be called s^U and s^U's ephemeral private key will be denoted by $\tilde{u}$. The simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*($\hat{A}, \hat{B}$): $\mathcal{S}$ executes Step 1 of the protocol. However, if the session being created is s^U, then $\mathcal{S}$ deviates by setting the ephemeral public key X to be U.

2. *Send*$(\hat{B}, \hat{A}, X)$: $\mathcal{S}$ executes Step 2 of the protocol. However, if the session being created is s^U, $\mathcal{S}$ deviates by setting the ephemeral public key Y to be U. In addition if $\hat{B} = \hat{V}$ or $Y = U$, then $\mathcal{S}$ sets $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.

3. *Send*$(\hat{A}, \hat{B}, X, Y)$: $\mathcal{S}$ executes Step 3 of the protocol. However, if $\hat{A} = \hat{V}$ or $X = U$, then $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$.

4. *EphemeralKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

5. *SessionKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

6. *StaticKeyReveal*$(\hat{A})$: $\mathcal{S}$ responds to the query faithfully, unless $\hat{A} = \hat{V}$ in which case $\mathcal{S}$ aborts with failure.

7. *Establish*$(\hat{M})$: $\mathcal{S}$ responds to the query faithfully.

8. $\mathtt{H}_1(x, a)$: $\mathcal{S}$ simulates a random oracle in the usual way except if $\hat{A}$ owns s^U and $x = \tilde{u}$, in which case $\mathcal{S}$ aborts with failure.

9. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.

10. $\mathtt{H}(\sigma, X, Y, \hat{A}, \hat{B})$:

    (a) If $X = U$ and $\hat{B} = \hat{V}$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$.
        i. If $\tau = 1$, then $\mathcal{S}$ computes $\Pi = \sigma Y^{-a\mathrm{D}} V^{-a\mathrm{DE}} = g^{uv\mathrm{E}+uy}$.
        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

    (b) If $Y = U$ and $\hat{A} = \hat{V}$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$
        i. If $\tau = 1$, then $\mathcal{S}$ computes $\Pi = \sigma X^{-b\mathrm{E}} V^{-b\mathrm{DE}} = g^{uv\mathrm{D}+uy}$.
        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

    (c) If $\sigma \neq \xi(XA^{\mathrm{D}}, YB^{\mathrm{E}})$ and either $U \in \{X, Y\}$ or $\hat{V} \in \{\hat{A}, \hat{B}\}$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, YB^{\mathrm{E}}, \sigma)$.
        i. If $\tau = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, YB^{\mathrm{E}}), X, Y, \hat{A}, \hat{B}\right)$.
        ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

    (d) $\mathcal{S}$ simulates a random oracle in the usual way.

11. *Test*(s^t): If $\mathrm{s}^t \neq \mathrm{s}^U$ or the peer of s^t is not $\hat{V}$, then $\mathcal{S}$ aborts with failure; otherwise responds to the query faithfully.

12. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $\overline{T_m} \wedge \overline{DL} \wedge M^*$. The simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. The probability that the test session has peer $\hat{V}$ and outgoing ephemeral public key U is at least $\frac{1}{n^2 s}$. Suppose that this is indeed the case, so $\mathcal{S}$ does not abort as in Step 11, and suppose that event $\overline{T_m}$ occurs. Since $\tilde{u}$ is used only in the test session, $\mathcal{M}$ must obtain it via an *EphemeralKeyReveal* query before making an $\mathtt{H}_1$ query that includes $\tilde{u}$. Under event $\overline{DL}$, the adversary first issues a *StaticKeyReveal* query to a party before making an $\mathtt{H}_1$ query that includes that party's static private key. Since the test session is fresh, and s^t has no matching session a successful $\mathcal{M}$ does not query for x; hence $\mathcal{S}$ does not abort as described in Steps 6 and 8.

Without loss of generality let Y denote the incoming ephemeral public key selected by $\mathcal{M}$ for the test session $\mathrm{S}^t = (\hat{A}, \hat{V}, U, Y)$. Under event M^*, $\mathcal{M}$ queries $\mathtt{H}$ with $(\sigma, U, Y, \hat{A}, \hat{V})$ where $\mathrm{DDH}(UA^{\mathrm{D}}, YV^{\mathrm{E}}, \sigma) = 1$, in which case as described in Step 10(a)i $\mathcal{S}$ computes

$$\Pi = \sigma Y^{-a\mathrm{D}} V^{-a\mathrm{DE}} = g^{uv\mathrm{E}+uy}.$$

Without the knowledge of $y = \log_g Y$, $\mathcal{S}$ is unable to compute $\mathrm{CDH}(U, V)$. Following the Forking Lemma [20, Lemma 2] approach, $\mathcal{S}$ runs $\mathcal{M}$ on the same input and the same coin flips but with carefully modified answers to the $\mathtt{H}_2$ queries. Note that $\mathcal{M}$ must have queried $\mathtt{H}_2$ with $(Y, \hat{A}, \hat{B})$ in its first run, because otherwise $\mathcal{M}$ would be unable to compute σ except with negligible probability. For the second run of $\mathcal{M}$, $\mathcal{S}$ responds to $\mathtt{H}_2(Y, \hat{A}, \hat{B})$ with a value $\mathrm{E}' \neq \mathrm{E}$ selected uniformly at random. Another way of describing the second run is: $\mathcal{M}$ is rewound to the point where $\mathcal{M}$ queries $\mathtt{H}_2$ with $(Y, \hat{A}, \hat{B})$ and the query is answered with a random value E' different from E. If $\mathcal{M}$ succeeds in the second run, in Step 10(a)i $\mathcal{S}$ computes

$$\Pi' = \sigma' Y^{-a\mathrm{D}'} V^{-a\mathrm{D}'\mathrm{E}'} = g^{uv\mathrm{E}'+uy}$$

and thereafter obtains

$$\mathrm{CDH}(U, V) = \left(\frac{\Pi}{\Pi'}\right)^{(\mathrm{E}-\mathrm{E}')^{-1}}.$$

The forking is at the expense of introducing a wider gap in the reduction. The success probability of $\mathcal{S}$, excluding negligible terms, is

$$\Pr(\mathcal{S}) \geq \frac{1}{s}\frac{1}{n^2}\frac{C}{h_2} p_{\tilde{m}} \tag{3}$$

where C is a constant arising from the use of the Forking Lemma[3]

4.3 Analysis

Suppose that event M occurs. Combining Equations (1), (2) and (3), the success probability of $\mathcal{S}$ is

$$\Pr(\mathcal{S}) \geq \max\left\{\frac{1}{n(\lambda)} p_{DL}(\lambda), \frac{2}{(n(\lambda)s(\lambda))^2} p_m(\lambda), \frac{C}{s(\lambda)n(\lambda)^2 h_2(\lambda)} p_{\tilde{m}}(\lambda)\right\}, \tag{4}$$

which is non-negligible in λ.

The simulation requires $\mathcal{S}$ to perform group exponentiations, access the DDH oracle, and simulate random oracles. Since $q = \Theta(2^\lambda)$, a group exponentiation takes time $\mathcal{T}_{\mathcal{G}} = \mathcal{O}(\lambda)$ group multiplications. Assume that a DDH oracle call takes time $\mathcal{T}_{\mathrm{DDH}} = \mathcal{O}(\lambda)$. Responding to an $\mathtt{H}$ query takes time $\mathcal{T}_{\mathtt{H}} = \mathcal{O}(\lambda)$; similarly, responding to $\mathtt{H}_1$ and $\mathtt{H}_2$ queries takes time $\mathcal{T}_{\mathtt{H}_1}(\lambda)$ and $\mathcal{T}_{\mathtt{H}_2}(\lambda)$. Taking the largest times from among all simulations for answering $\mathcal{M}$'s queries, the running time of $\mathcal{S}$ is bounded by

$$\mathcal{T}_{\mathcal{S}} \leq (\mathcal{T}_{3\mathcal{G}} + (\mathcal{T}_{\mathrm{DDH}} + 2\mathcal{T}_{\mathcal{G}} + \mathcal{T}_{\mathtt{H}}) + (\mathcal{T}_{\mathcal{G}} + \mathcal{T}_{\mathtt{H}_1}) + \mathcal{T}_{\mathtt{H}_2})\,\mathcal{T}_{\mathcal{M}}. \tag{5}$$

Thus, if $\mathcal{M}$ is polynomially bounded, then there is an algorithm $\mathcal{S}$ that succeeds in solving the GDH problem in $\mathcal{G}$ with non-negligible probability. Furthermore $\mathcal{S}$ runs in polynomial time, contradicting the GDH assumption in $\mathcal{G}$. This concludes the proof of Theorem 4.1. □

[3]The constant C in Pointcheval and Stern's version of the lemma is 84480^{-1}. Its value has not been worked out in Theorem 4.1's (and Krawzcyk's) use of the Forking Lemma.

5 One-pass CMQV

One-pass protocols are useful in environments, where the responder is not available for immediate reply; hence there are fundamental differences with interactive environments. There are important scenarios, such as email, where one-pass protocols are useful and therefore they are worth studying.

5.1 Description

In a nutshell, one-pass CMQV is two-pass CMQV, where the ephemeral public key Y of the responder is the identity element in the group. To that end there is no need to include Y in the key derivation.

Definition 5.1 (one-pass CMQV) *The protocol proceeds as follows:*

1. *Upon activation $(\hat{A}, \hat{B})$, party $\hat{A}$ (the initiator) performs the steps:*
   (a) *Select an ephemeral private key $\tilde{x} \in_R \{0,1\}^\lambda$.*
   (b) *Compute the ephemeral public key $X = g^{\mathtt{H}_1(\tilde{x},a)}$.*
   (c) *Compute $\mathrm{D} = \mathtt{H}_2(X, \hat{A}, \hat{B})$ and $\sigma = B^{\mathtt{H}_1(x,a)+\mathrm{D}a}$.*
   (d) *Compute $\kappa = \mathtt{H}(\sigma, X, \hat{A}, \hat{B})$ and destroy x and σ.*
   (e) *Send $(\hat{B}, \hat{A}, X)$ to $\hat{B}$ and complete session $\mathrm{S} = (\hat{A}, \hat{B}, X)$ with session key κ.*
2. *Upon activation $(\hat{B}, \hat{A}, X)$, party $\hat{B}$ (the responder) performs the steps:*
   (a) *Verify that $X \in \mathcal{G}^*$.*
   (b) *Compute $\mathrm{D} = \mathtt{H}_2(X, \hat{A}, \hat{B})$ and $\sigma = (XA^{\mathrm{D}})^b$.*
   (c) *Compute $\kappa = \mathtt{H}(\sigma, X, \hat{A}, \hat{B})$ and destroy σ.*
   (d) *Complete session $\mathrm{S} = (\hat{B}, \hat{A}, X)$ with session key κ.*

If any verification fails, then the party erases all session specific information from its memory and aborts the session.

5.2 One-pass model modifications

Even though the definition of secure protocol (Definition 2.2) does not depend on the number of protocol flows, the definition of fresh session has to be modified to fit the needs of a one-pass protocol. In particular, one-pass protocols cannot achieve forward secrecy since an adversary can compute a session key by learning the static private key of the responder.

Definition 5.2 (one-pass fresh session) *Let* S *be the session identifier of a completed session, owned by an honest party $\hat{A}$ with intended peer $\hat{B}$, who is also honest. Let* S^* *be the session identifier of the matching session of* S*, if it exists. Define* S *to be* fresh *if none of the following conditions hold:*

*(i) $\mathcal{M}$ issues a SessionKeyReveal(*S*) query or a SessionKeyReveal(*S^**) query (if* S^* *exists);*

(ii) if $\hat{A}$ is the initiator then $\mathcal{M}$ makes either of the following queries:

- *both StaticKeyReveal($\hat{A}$) and EphemeralKeyReveal(*S*), or*

– *StaticKeyReveal($\hat{B}$);*

(iii) if $\hat{A}$ is the responder then $\mathcal{M}$ makes either of the following queries

– *StaticKeyReveal($\hat{A}$) or*

– *StaticKeyReveal($\hat{B}$).*

We point out that by replaying messages from $\hat{A}$ to $\hat{B}$ an adversary $\mathcal{M}$ could force multiple sessions owned by $\hat{B}$ sharing the same session key κ. Let S_κ be the set of sessions owned by $\hat{B}$ with the same session key κ. Since all sessions in S_κ have the same session identifiers, $\mathcal{M}$ cannot compromise a single session in S_κ without compromising all sessions in S_κ. Therefore, the definition of session identifier accounts for replay attacks.

6 One-pass CMQV security

The security argument for one-pass CMQV is very similar to the security argument for two-pass CMQV Section 4.

Theorem 6.1 *If* $\mathtt{H}_1$,$\mathtt{H}_2$ *and* $\mathtt{H}$ *are random oracles, and $\mathcal{G}$ is a group where the GDH assumption holds, then one-pass CMQV is eCK secure.*

Proof: Verifying condition 1 of Definition 2.2 is straightforward; it remains to verify condition 2.

Let λ denote the security parameter, whence $q = |\mathcal{G}| = \Theta(2^\lambda)$. Let $\mathcal{M}$ be a polynomially (in λ) bounded one-pass CMQV adversary. The adversary $\mathcal{M}$ is said to be successful (event M) with non-negligible probability if $\mathcal{M}$ wins the distinguishing game described in Section 2 with probability $\frac{1}{2} + p(\lambda)$, where $p(\lambda)$ is non-negligible. Assume that $\mathcal{M}$ operates in an environment that involves at most $n(\lambda)$ parties, and within a party $\mathcal{M}$ activates at most $s(\lambda)$ sessions as the initiator within a party, and makes at most $h_1(\lambda), h_2(\lambda)$ and $h(\lambda)$ queries to oracles $\mathtt{H}_1$,$\mathtt{H}_2$ and $\mathtt{H}$, respectively; and terminates after time at most $\mathcal{T}_\mathcal{M}$. Let the test session be $\mathrm{s}^t = (\hat{A}, \hat{B}, X)$ and let H denote the event that $\mathcal{M}$ queries $\mathtt{H}$ with $(\sigma, X, \hat{A}, \hat{B})$, where $\sigma = \mathrm{CDH}(XA^{\mathrm{D}}, B)$. Let $\overline{H}$ be the complement of H and s^* be any completed session owned by an honest party, such that s^t and s^* are non-matching. Since s^t and s^* are non-matching, the input to the key derivation function $\mathtt{H}$ are different for s^t and s^*. And since $\mathtt{H}$ is a random oracle it follows that $\mathcal{M}$ cannot obtain any information about the test session key from the session key of non-matching sessions. Hence $\Pr(M \wedge \overline{H}) \leq \frac{1}{2}$ and

$$\Pr(M) = \Pr(M \wedge \overline{H}) + \Pr(M \wedge H) \leq \frac{1}{2} + \Pr(M \wedge H),$$

whence $\Pr(M \wedge H) \geq p$; henceforth the event $M \wedge H$ is denoted by M^*.

Following the standard approach, such an adversary $\mathcal{M}$ is used to construct a GDH solver $\mathcal{S}$ that succeeds with non-negligible probability. Let $\xi : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ be a random function known only to $\mathcal{S}$, such that $\xi(X, Y) = \xi(Y, X)$. The algorithm $\mathcal{S}$ will use ξ to simulate $\mathrm{CDH}(X, Y)$ when $\mathcal{S}$ may not know $\log_g(X)$ or $\log_g(Y)$. Let the input to the GDH challenge be (U, V) and consider the following complementary events:

DL. There exists an honest party $\hat{B}$ such that $\mathcal{M}$, during its execution, queries $\mathtt{H}_1$ with $(*, b)$, before issuing a *StaticKeyReveal*($\hat{B}$) query. (Note that $\mathcal{M}$ does not necessarily make a *StaticKeyReveal*($\hat{B}$) query.)

$\overline{DL}$. During its execution, for every honest party $\hat{B}$ for which $\mathcal{M}$ queries $\mathtt{H}_1$ with $(*, b)$, $\mathcal{M}$ issued *StaticKeyReveal*$(\hat{B})$ before the first $(*, b)$ query to $\mathtt{H}_1$.

If $\mathcal{M}$ succeeds with non-negligible probability, and hence $\Pr(M^*) \geq p$, it must be the case that either event $DL \wedge M^*$ or event $\overline{DL} \wedge M^*$ occurs with non-negligible probability. These events are considered separately.

6.1 Event DL

Simulation. Suppose that event $DL \wedge M^*$ occurs with non-negligible probability. In this case $\mathcal{S}$ prepares n parties. One party, called $\hat{V}$, is selected at random and assigned static public key V; $\mathcal{S}$ represents $\hat{V}$'s static private key by $\nu \in_R \mathbb{Z}_q$. The remaining $n-1$ parties are assigned random static key pairs. The adversary $\mathcal{M}$ is initiated on this set of parties and the simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*$(\hat{A}, \hat{B})$: $\mathcal{S}$ executes Step 1 of the protocol. However, if $\hat{A} = \hat{V}$, then $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, B)$.
2. *Send*$(\hat{B}, \hat{A}, X)$: $\mathcal{S}$ executes Step 2 of the protocol. However, if $\hat{B} = \hat{V}$, then $\mathcal{S}$ sets $\sigma = \xi(XA^{\mathrm{D}}, B)$.
3. *EphemeralKeyReveal*(S): $\mathcal{S}$ responds to the query faithfully.
4. *SessionKeyReveal*(S): $\mathcal{S}$ responds to the query faithfully.
5. *StaticKeyReveal*$(\hat{A})$: $\mathcal{S}$ responds to the query faithfully, unless $\hat{A} = \hat{V}$ in which case $\mathcal{S}$ aborts with failure.
6. *Establish*$(\hat{M})$: $\mathcal{S}$ responds to the query faithfully.
7. $\mathtt{H}_1(s, c)$: $\mathcal{S}$ checks if the equation $g^c = V$ holds, in which case $\mathcal{S}$ stops $\mathcal{M}$ and is successful by outputting $\mathrm{CDH}(U, V) = U^c$; otherwise $\mathcal{S}$ simulates a random oracle in the usual way.
8. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.
9. $\mathtt{H}(\sigma, X, \hat{A}, \hat{B})$:
   (a) If $\hat{V} \in \{\hat{A}, \hat{B}\}$ and $\sigma \neq \xi(XA^{\mathrm{D}}, B)$, then $\mathcal{S}$ obtains $\tau = \mathrm{DDH}(XA^{\mathrm{D}}, B, \sigma)$.
      i. If $\tau = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, B), X, \hat{A}, \hat{B}\right)$.
      ii. If $\tau = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.
   (b) $\mathcal{S}$ simulates a random oracle in the usual way.
10. *Test*(S): $\mathcal{S}$ responds to the query faithfully.
11. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $DL \wedge M^*$. $\mathcal{S}$'s simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. With probability at least $\frac{1}{n}$, $\mathcal{S}$ assigns the public key V to an honest party $\hat{B}$ for whom $\mathcal{M}$ will query $\mathtt{H}_1(*, b)$ without first issuing a *StaticKeyReveal*$(\hat{B})$ query. In this case $\mathcal{S}$ is successful as described in Step 7 and the abortions as in Steps 5 and 11 do not occur. Hence if event $DL \wedge M^*$ occurs with probability p_{DL}, then $\mathcal{S}$ is successful with probability $\Pr(S)$ that is bounded by

$$\Pr(\mathcal{S}) \geq \frac{1}{n} p_{DL}. \tag{6}$$

6.2 Event $\overline{DL}$

Let T_m be the event "the test session has a matching session owned by an honest party or the test session owner is also the session initiator". Event $\overline{DL} \wedge M^*$ is further subdivided into the following complementary events:

(i) $T_m = (\overline{DL} \wedge M^* \wedge T_m)$, and

(ii) $\overline{T_m} = (\overline{DL} \wedge M^* \wedge \overline{TM})$.

Let $p_{\tilde{DL}} = \Pr(\overline{DL} \wedge M^*)$, $p_m = \Pr(T_m)$, and $p_{\tilde{m}} = \Pr(\overline{T_m})$. Since T_m and $\overline{T_m}$ are complementary, $p_{\tilde{DL}} = p_m + p_{\tilde{m}}$. Therefore, if event $\overline{DL} \wedge M^*$ occurs with non-negligible probability, then either T_m or $\overline{T_m}$ occurs with non-negligible probability. Events T_m and $\overline{T_m}$ are next considered separately.

6.2.1 Event T_m

Simulation in event T_m. Suppose that event T_m occurs with non-negligible probability. In this case $\mathcal{S}$ establishes n parties. One party, called $\hat{V}$, is selected at random and assigned static public key V; $\mathcal{S}$ represents $\hat{V}$'s static private key by $\nu \in_R \mathbb{Z}_q$. The remaining $n-1$ parties are assigned random static key pairs. Furthermore, $\mathcal{S}$ randomly selects an integer $i \in_R [1, \ldots, ns]$. The i'th session created will be called s^U and s^U's ephemeral private key will be denoted by $\tilde{u}$. The simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*$(\hat{A}, \hat{B})$: $\mathcal{S}$ executes Step 1 of the protocol. However, if the session being created is s^U, $\mathcal{S}$ deviates by setting the ephemeral public key X to be U. In addition, if $X = U$ or $\hat{A} = \hat{V}$, then $\mathcal{S}$ sets $\sigma = \xi(XA^{\mathrm{D}}, B)$.

2. *Send*$(\hat{B}, \hat{A}, X)$: $\mathcal{S}$ executes Step 2 of the protocol. However, if $\hat{B} = \hat{V}$, then $\mathcal{S}$ deviates by setting $\sigma = \xi(XA^{\mathrm{D}}, B)$.

3. *EphemeralKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

4. *SessionKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

5. *StaticKeyReveal*$(\hat{A})$: $\mathcal{S}$ responds to the query faithfully, unless $\hat{A} = \hat{V}$, in which case $\mathcal{S}$ aborts with failure.

6. *Establish*$(\hat{M})$: $\mathcal{S}$ responds to the query faithfully.

7. $\mathtt{H}_1(x, a)$: $\mathcal{S}$ simulates a random oracle in the usual way except if $\hat{A}$ owns s^U and $x = \tilde{u}$ or if $\hat{A}$ owns s^V and $x = \tilde{v}$, in which case $\mathcal{S}$ aborts with failure.

8. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.

9. $\mathtt{H}(\sigma, X, \hat{A}, \hat{B})$:

   (a) If $X = U$, $\hat{B} = \hat{V}$ and $\mathrm{DDH}(XA^{\mathrm{D}}, B, \sigma) = 1$, then $\mathcal{S}$ aborts $\mathcal{M}$ and is successful by outputting $\mathrm{CDH}(U, V) = \sigma V^{-a^{\mathrm{D}}}$.

   (b) If $\sigma \neq \xi(XA^{\mathrm{D}}, Y)$ and either $\hat{V} \in \{\hat{A}, \hat{B}\}$ or $X = U$, then

      i. if $\mathrm{DDH}(XA^{\mathrm{D}}, B, \sigma) = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, Y), X, \hat{A}, \hat{B}\right)$;

      ii. if $\mathrm{DDH}(XA^{\mathrm{D}}, B, \sigma) = 1$, then $\mathcal{S}$ simulates a random oracle in the usual way.

(c) $\mathcal{S}$ simulates a random oracle in the usual way.

10. *Test*(s^t): If the peer of s^U is not $\hat{V}$ or if s^t is neither s^U nor the session matching to s^U, then $\mathcal{S}$ aborts; otherwise responds to the query faithfully.

11. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $T_m \wedge \overline{DL} \wedge M^*$. $\mathcal{S}$'s simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. The probability that $\mathcal{M}$ selects $\hat{V}$ as s^U's peer and s^t is either s^U or its matching session is at least $\frac{2}{ns^2}$. Suppose that this is the case, so $\mathcal{S}$ does not abort as in Step 10, and suppose that event T_m occurs. Without loss of generality, let $\mathrm{s}^U = (\hat{A}, \hat{V}, U)$. Since $\tilde{u}$ is used only in s^U, $\mathcal{M}$ must obtain $\tilde{u}$ via an *EphemeralKeyReveal* query before making an $\mathtt{H}_1$ query that includes $\tilde{u}$. Under event $\overline{DL}$, the adversary first issues a *StaticKeyReveal*($\hat{A}$) query before making an $\mathtt{H}_1$ query that includes x. Since the test session is fresh, $\mathcal{S}$ does not abort as described in Step 5 and 7. Under event M^*, except with negligible probability of guessing $\xi(UA^{\mathrm{D}}, V)$, $\mathcal{S}$ is successful as described in Step 9a and does not abort as in Step 11. Therefore if event T_m occurs, then the success probability of $\mathcal{S}$ is

$$\Pr(\mathcal{S}) \geq \frac{2}{ns^2} p_m. \tag{7}$$

6.2.2 Event $\overline{T_m}$

Simulation. Suppose that event $\overline{T_m}$ occurs with non-negligible probability, in which case no honest party owns a session matching to the test session and the test session owner is also the responder. In this case $\mathcal{S}$ prepares n parties. Two of these parties, denoted by $\hat{U}$ and $\hat{V}$, are selected uniformly at random and assigned static public keys U and V, respectively. The remaining $n-2$ parties are assigned random static key pairs. The algorithm $\mathcal{S}$ will use $\upsilon \in_R \mathbb{Z}_q$ and $\nu \in_R \mathbb{Z}_q$, to represent the static private keys of $\hat{U}$ and $\hat{V}$, respectively. The simulation of $\mathcal{M}$'s environment proceeds as follows:

1. *Send*($\hat{A}, \hat{B}$): $\mathcal{S}$ executes Step 1 of the protocol. However, if $\hat{A} \in \{\hat{U}, \hat{V}\}$, then $\mathcal{S}$ sets $\sigma = \xi(XA^{\mathrm{D}}, B)$.

2. *Send*($\hat{B}, \hat{A}, X$): $\mathcal{S}$ executes Step 2 of the protocol. However, if $\hat{B} \in \{\hat{U}, \hat{V}\}$, then $\mathcal{S}$ sets $\sigma = \xi(XA^{\mathrm{D}}, B)$.

3. *EphemeralKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

4. *SessionKeyReveal*(s): $\mathcal{S}$ responds to the query faithfully.

5. *StaticKeyReveal*($\hat{A}$): $\mathcal{S}$ responds to the query faithfully, unless $\hat{A} \in \{\hat{U}, \hat{V}\}$ in which case $\mathcal{S}$ aborts with failure.

6. *Establish*($\hat{M}$): $\mathcal{S}$ responds to the query faithfully.

7. $\mathtt{H}_1(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.

8. $\mathtt{H}_2(*)$: $\mathcal{S}$ simulates a random oracle in the usual way.

9. $\mathtt{H}(\sigma, X, \hat{A}, \hat{B})$:

   (a) If $\{\hat{A}, \hat{B}\} = \{\hat{U}, \hat{V}\}$ and $\mathrm{DDH}(XA^{\mathrm{D}}, B, \sigma) = 1$, then $\mathcal{S}$ records σ.

(b) If $\sigma \neq \xi(XA^{\mathrm{D}}, B)$ and either $\hat{U} \in \{\hat{A}, \hat{B}\}$ or $\hat{V} \in \{\hat{A}, \hat{B}\}$, then

i. if $\mathrm{DDH}(XA^{\mathrm{D}}, B) = 1$, then $\mathcal{S}$ returns $\mathtt{H}\left(\xi(XA^{\mathrm{D}}, B), X, \hat{A}, \hat{B}\right)$;

ii. if $\mathrm{DDH}(XA^{\mathrm{D}}, B) = 0$, then $\mathcal{S}$ simulates a random oracle in the usual way.

(c) $\mathcal{S}$ simulates a random oracle in the usual way.

10. *Test*(s^t): If the communicating partners of s^t are not $\hat{U}$ and $\hat{V}$, then $\mathcal{S}$ aborts with failure; otherwise responds to the query faithfully.

11. $\mathcal{M}$ outputs a guess: $\mathcal{S}$ aborts with failure.

Analysis of event $\overline{T_m} \wedge \overline{DL} \wedge M^*$. The simulation of $\mathcal{M}$'s environment is perfect except with negligible probability. The probability that $\hat{U}$ and $\hat{V}$ are the test session's communicating partners is at least $\frac{2}{n^2}$. Suppose that this is indeed the case, so $\mathcal{S}$ does not abort as in Step 10, and suppose that event $\overline{T_m}$ occurs. Since the test session is fresh, and s^t has no matching session, then $\mathcal{S}$ does not abort as described in Step 5.

Without loss of generality, let $\mathrm{s}^t = (\hat{U}, \hat{V}, B)$, where B denotes s^t's incoming ephemeral public key selected by $\mathcal{M}$. Under event M^*, except with negligible probability of guessing $\xi(YU^{\mathrm{D}}, V)$, $\mathcal{M}$ queries $\mathtt{H}$ with $(\sigma, Y, \hat{U}, \hat{V})$, where $\mathrm{DDH}(YU^{\mathrm{D}}, V, \sigma) = 1$, in which case as described in Step 9a, $\mathcal{S}$ obtains

$$\sigma = g^{uv\mathrm{D}+vy}.$$

Without knowledge of $y = \log_g Y$, $\mathcal{S}$ is unable to compute $\mathrm{CDH}(U, V)$. Following the Forking Lemma [20] approach, $\mathcal{S}$ runs $\mathcal{M}$ on the same input and the same coin flips but with carefully modified answers to the $\mathtt{H}_2$ queries. Note that $\mathcal{M}$ must have queried $\mathtt{H}_2$ with $(Y, \hat{A}, \hat{B})$ in its first run, because otherwise $\mathcal{M}$ would be unable to compute σ except with negligible probability. For the second run of $\mathcal{M}$, $\mathcal{S}$ responds to $\mathtt{H}_2(Y, \hat{A}, \hat{B})$ with a value $\mathrm{D}' \neq \mathrm{D}$ selected uniformly at random. Another way of describing the second run is: $\mathcal{M}$ is rewound to the point where $\mathcal{M}$ queries $\mathtt{H}_2$ with $(Y, \hat{A}, \hat{B})$ and the query is answered with a random value D' different from D. If $\mathcal{M}$ succeeds in the second run, in Step 9a $\mathcal{S}$ obtains

$$\sigma' = g^{uv\mathrm{D}'+vy}$$

and thereafter obtains

$$\mathrm{CDH}(U, V) = \left(\frac{\sigma}{\sigma'}\right)^{(\mathrm{D}-\mathrm{D}')^{-1}}.$$

The forking is at the expense of introducing a wider gap in the reduction. The success probability of $\mathcal{S}$, excluding negligible terms, is

$$\Pr(\mathcal{S}) \geq \frac{1}{n^2}\frac{C}{h_2}p_{\tilde{m}}, \tag{8}$$

where C is a constant arising from the Forking Lemma.

6.3 Analysis

Suppose that event M occurs. Combining Equations (6), (7) and (8), the success probability of $\mathcal{S}$ is

$$\Pr(\mathcal{S}) \geq \max\left\{\frac{1}{n(\lambda)}p_{DL}(\lambda), \frac{2}{n(\lambda)s(\lambda)^2}p_m(\lambda), \frac{C}{n(\lambda)^2 h_2(\lambda)}p_{\tilde{m}}(\lambda)\right\}, \tag{9}$$

which is non-negligible in λ.

The simulation requires $\mathcal{S}$ to perform group exponentiations, access the DDH oracle, and simulate random oracles. Since $q = \Theta(2^\lambda)$, a group exponentiation takes time $\mathcal{T}_\mathcal{G} = \mathcal{O}(\lambda)$ group multiplications. Assume that a DDH oracle call takes time $\mathcal{T}_{\text{DDH}} = \mathcal{O}(\lambda)$. Responding to an $\mathtt{H}$ query takes time $\mathcal{T}_\mathtt{H} = \mathcal{O}(\lambda)$; similarly responding to $\mathtt{H}_1$ and $\mathtt{H}_2$ queries takes time $\mathcal{T}_{\mathtt{H}_1}(\lambda)$ and $\mathcal{T}_{\mathtt{H}_2}(\lambda)$, respectively. Taking the largest times from among all simulations for answering $\mathcal{M}$'s query, the running time of $\mathcal{S}$ is bounded by

$$\mathcal{T}_\mathcal{S} \leq (\mathcal{T}_{2\mathcal{G}} + (\mathcal{T}_{\text{DDH}} + \mathcal{T}_\mathcal{G} + \mathcal{T}_\mathtt{H}) + (\mathcal{T}_\mathcal{G} + \mathcal{T}_{\mathtt{H}_1}) + \mathcal{T}_{\mathtt{H}_2})\mathcal{T}_\mathcal{M}. \tag{10}$$

Thus, if $\mathcal{M}$ is polynomially bounded, then there is an algorithm $\mathcal{S}$ that succeeds in solving the GDH problem in $\mathcal{G}$ with non-negligible probability. Furthermore $\mathcal{S}$ runs in polynomial time, contradicting the GDH assumption in $\mathcal{G}$. This concludes the proof of Theorem 6.1. □

7 Comments

7.1 Model description

As described in [12] the adversary is not given an *Establish* query. Instead the adversary selects the identifiers for the parties and is allowed to register adversary controlled parties at the onset of the protocol, more precisely during the registration phase. The *Establish* query used here achieves the same if the first queries performed by the adversary are all *Establish* queries. Furthermore, it allows the adversary to adaptively select public keys and identities based on the ephemeral public keys selected by the honest parties. As such the query allows modeling of Kaliski [8] type unknown key share attacks.

7.2 Simulation

We stress that the eCK model allows for sessions where the initiator and the responder are the same party. As pointed out in Section 3.1 the security arguments have been carried out under the assumption that a party does not execute the protocol with itself. That assumption is needed in simulation Step 10(a)i of Section 4.2.2 and simulation Step 9a of Section 6.2.1, where it is *assumed* that $\mathcal{S}$ possesses the value a. If the owner and the peer of the test session is the same party $\hat{V}$, then algorithm $\mathcal{S}$ does not posses the value a and cannot perform the required computations. Furthermore, simulation Step 10 of Section 6.2.2 requires distinct test session partners. These shortcoming can be addressed by introducing additional events and utilizing the "gap square Diffie-Hellman assumption". For sake of simplicity this has not been done.

8 Concluding remarks

The paper presented CMQV, a modification of the MQV key agreement protocol. On the positive side the new protocol is secure in the extended Canetti-Krawzcyk model. Moreover it achieves the performance of the original MQV protocol, and has intuitive design principles and a relatively simple security proof. On the negative side the reduction argument is not tight, in particular the Forking Lemma appears to be essential for the security argument. It remains to be seen if there exists a protocol that achieves the performance of MQV and at the same time enjoys a security reduction that is as tight as the security reduction for NAXOS.

References

[1] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone. Validation of elliptic curve public keys. In Y. G. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *LNCS*, pages 211–223. Springer Verlag, 2003.

[2] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289, Santa Barbara, CA, USA, 2004. Springer Verlag.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155, Bruges, Belgium, 2000. Springer Verlag.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 232–249, Santa Barbara, CA, USA, 1993. Springer Verlag. Full version available at `http://www.cs.ucdavis.edu/~rogaway/papers/eakd-abstract.html`.

[5] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In H. Imai and Y. Zheng, editors, *Public Key Cryptography – PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer Verlag, March 1999.

[6] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474, Aarhus, Denmark, 2001. Springer Verlag. Full version available at `http://eprint.iacr.org/2001/040/`.

[7] K.-K. R. Choo, C. Boyd, and Y. Hitchcock. Examining indistinguishability-based proof models for key establishment protocols. In B. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 585–604. Springer Verlag, 2005.

[8] B. S. Kaliski Jr. An unknown key-share attack on the mqv key agreement protocol. *ACM Transaction on Information and System Security*, 4(3):275–288, 2001.

[9] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In R. Cramer, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer Verlag, 2005. Full version available at `http://eprint.iacr.org/2005/176`.

[10] H. Krawczyk. HMQV in IEEE P1263, July 2006. submission to the IEEE P1263 working group `http://grouper.ieee.org/groups/1363/P1363-Reaffirm/submissions/krawczyk-hmqv-spec.pdf`.

[11] S. Kunz-Jacques and D. Pointcheval. About the security of MTI/C0 and MQV. In R. De Prisco and M. Yung, editors, *Security and Cryptography for Networks – 5th International Conference, SCN 2006*, volume 4116 of *LNCS*, pages 156–172. Springer Verlag, September 2006.

[12] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In W. Susilo, J. K. Liu, and Y. Mu, editors, *Provable Security: First International Conference, ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16, Wollongong, Australia, 2007. Springer Verlag.

[13] K. Lauter and A. Mityagin. Security analysis of KEA authenticated key exchange protocol. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *LNCS*, pages 378–394. Springer Verlag, 2006.

[14] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.

[15] C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In B. S. Kaliski, Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *LNCS*, pages 249–263, Santa Barbara, CA, USA, 1997. Springer Verlag.

[16] A. Menezes. Another look at HMQV. *Journal of Mathematical Cryptology*, 1(1):47–64, 2007.

[17] A. Menezes and B. Ustaoglu. On the importance of public-key validation in the MQV and HMQV key agreement protocols. In R. Barua and T. Lange, editors, *Progress in Cryptology – INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 133–147. Springer Verlag, 2006.

[18] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, 1997.

[19] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In K. Kim, editor, *Public Key Cryptography – PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer Verlag, 2001.

[20] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer Verlag, May 1996.

[21] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[22] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008.

Errata

Known deficiencies found in the short version [22].

Model description: In the short version of this paper a session identifier s is given by $(\hat{A}, \hat{B}, X, Y)$ with matching session s^* given by $(\hat{B}, \hat{A}, Y, X)$. These identifiers differ from the original eCK definition: they do not carry information about the role of session owners and hence both parties $\hat{A}$ and $\hat{B}$ may view themselves as initiators. If both parties perform the protocol as initiators, then sessions s and s^* even though matching do not compute the same session key, since the order of identities and public keys used in the key derivation function is reversed. The updated session identifier definition, which is the original eCK definition, rules this possibility out.
Credit: discussions with Cas Cremers and Alfred Menezes.

Efficiency: Table 2, in the short version of this paper, had an incorrect estimate of Shamir's trick [18, Algorithm 14.88]; it has been addressed in the current version.
Credit: discussions with Jiang Wu.

Revisions

June 22, 2009

- Updated the eCK model description.
- Added details to the security argument for two-pass CMQV (Theorem 4.1).
- Added security argument for one-pass CMQV (Theorem 6.1).
- Updated Table 2, modified Figure 1 and reworded Definitions 3.1 and 5.1.
- Added Section 7.
- Corrected references and updated contact information.