Authenticated Key Exchange Protocols Based on Factoring Assumption

Hai Huang

Department of Computer Science and Engineering,
Zhejiang Sci-Tech University, Hangzhou, 310018, China
haihuang1005@gmail.com

Abstract. This paper investigates authenticated key exchange protocols over signed quadratic residues group $\mathbb{QR}_N^+$, which is originally used for encryption schemes. The key technical tool developed by Hofheinz et al. is that in group $\mathbb{QR}_N^+$ the strong Diffie-Hellman (SDH) problem is implied by the factoring assumption.

To apply group $\mathbb{QR}_N^+$ to authenticated key exchange protocols in the enhanced Canetti-Krawczyk (eCK) model, we extend Hofheinz et al.'s technique and introduce a new proof approach called k-th power.

The k-th power proof approach is almost generic, i.e., applying it to many, if not all, existing authenticated Diffie-Hellman key exchange protocols in eCK model under gap assumption immediately produces protocols in eCK model under factoring assumption if they work over $\mathbb{QR}_N^+$.

As one application of k-th power approach, we show that FS protocol, in which k is a constant, is provably secure in eCK model under factoring assumption if it works over $\mathbb{QR}_N^+$.

Our technique also applies to other protocols, e.g., UP,HMQV and its variants, in which k is a non-constant, but at the cost of degrading a factor in the reduction.

Keywords: Authenticated key exchange, Factoring assumption, $\mathbb{QR}_N^+$, eCK model.

1 Introduction

Key exchange protocols enable two parties, Alice (A) and Bob (B), to establish a shared session key via an unsecured channel. The classic Diffie-Hellman (DH) key exchange protocol is as follows: Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q. The exchange messages are $X = g^x, Y = g^y$ and the final session key is usually of the form $\mathrm{H}(g^{xy})$. It is well known that DH protocol is only secure against a passive attacker and vulnerable to the active man-in-the-middle attacker. Subsequently, a lot of work has been dedicated to the design of *authenticated* key exchange protocols which are secure against the active attacker.

1.1 Signed Quadratic Residues

Hofheinz et al. introduced a group called signed quadratic residues ($\mathbb{QR}_N^+$). The group is useful for cryptography because it is a "gap group", in which the

W. Susilo and R. Reyhanitabar (Eds.): ProvSec 2013, LNCS 8209, pp. 21–37, 2013.

membership in the group can be publicly verified while the computational problem, i.e., computing square roots, is equivalent to factoring N. Then, they showed that in the $\mathbb{QR}_N^+$ group, the strong Diffie-Hellman (SDH) problem [1] is implied by the factoring assumption.

As one application of $\mathbb{QR}_N^+$ group, Hofheinz et al. re-analyzed the Hybrid ElGamal encryption scheme which was originally over prime order subgroups of $\mathbb{Z}_p^*$, and showed that the Hybrid ElGamal over $\mathbb{QR}_N^+$ is CCA secure in random oracle model under factoring assumption. Since the security proof of Hybrid ElGamal does not use knowledge about the order of underlying group, the scheme itself remains unchanged.

1.2 Problems with Authenticated Key Exchange Protocols over $\mathbb{QR}_N^+$ Group

It is natural to ask whether $\mathbb{QR}_N^+$ group can be used to design authenticated key exchange protocols under factoring assumption, especially in a strong security model, e.g., eCK model. A natural example is as follows. The key derivation function is

$$k = H(Z_1, Z_2, Z_3, Z_4, X, Y, \hat{A}, \hat{B}), \text{ where } Z_1 = B^a, Z_2 = Y^a, Z_3 = B^x, Z_4 = Y^x \tag{1}$$

The protocol is clearly provably secure in eCK model under SDH assumption if the underlying group is a cyclic group of prime order q. On the other hand, the scheme is secure under factoring assumption if it works over $\mathbb{QR}_N^+$ group as the security proof does not use knowledge about the order of underlying group. However, the protocol requires 5 exponentiations which is unsatisfactory.

In the following, we provide some more efficient examples and their proof strategies in eCK model, and discuss the possibility of basing them on $\mathbb{QR}_N^+$ group. Assume that the owner of Test session is $\hat{A}$ and its peer is $\hat{B}$. Assume that the adversary can reveal the static private key of party $\hat{A}$ and no reveal query against the ephemeral private key of Test session is allowed, i.e, in the proofs the simulator sets the outgoing message of Test session to be $X = U$ and the static private key of the peer $B = V$, where (U, V) is a computational Diffie-Hellman (CDH) problem instance.

Example 1: HMQV protocol. The first example is HMQV protocol [10]. The key derivation function is

$$k = H(Z, X, Y, \hat{A}, \hat{B}), \text{ where } Z = (YB^e)^{x+ad}, e = h(Y, \hat{A}), d = h(X, \hat{B}) \tag{2}$$

In the security proof, with value Z, which is provided by the adversary, the simulator runs the adversary once again (forking lemma) and gets another Z'. Then, SIM computes

$$\left(\bar{Z} = \frac{Z/(YB^e)^{ad}}{Z'/(YB^{e'})^{ad}}\right)^{\frac{1}{(e-e')}} = \left(\frac{(YB^e)^x}{(YB^{e'})^x}\right)^{\frac{1}{(e-e')}} = \left(B^{x(e-e')}\right)^{\frac{1}{(e-e')}} = B^x = V^u \tag{3}$$

Example 2: FS protocol. The second example is FS protocol [4], which is most efficient one among FS protocol family (example 2, [4]). The key derivation function is

$$k = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B}), \quad \text{where} \quad Z_1 = (YB)^{x+a}, Z_2 = (YB^c)^{x+ac} \tag{4}$$

where c is a constant, e.g., $c = 2$. In the security proof, with Z_1, Z_2, which is provided by the adversary, SIM computes

$$\left(\bar{Z} = \frac{Z_2/(YB^c)^{ac}}{Z_1/(YB)^a}\right)^{\frac{1}{(c-1)}} = \left(\frac{(YB^c)^x}{(YB)^x}\right)^{\frac{1}{(c-1)}} = \left(B^{x(c-1)}\right)^{\frac{1}{(c-1)}} = B^x = V^u \tag{5}$$

Example 3: UP protocol. The third example is UP protocol [15]. The key derivation function is

$$k = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B}),$$
$$\text{where} \quad Z_1 = (YB^e)^{x+a}, Z_2 = (YB)^{x+ad}, e = h(Y), d = h(X) \tag{6}$$

In the security proof, with Z_1, Z_2, which is provided by the adversary, SIM computes

$$\left(\bar{Z} = \frac{Z_1/(YB^e)^a}{Z_2/(YB)^{ad}}\right)^{\frac{1}{(e-1)}} = \left(\frac{(YB^e)^x}{(YB)^x}\right)^{\frac{1}{(e-1)}} = \left(B^{x(e-1)}\right)^{\frac{1}{(e-1)}} = B^x = V^u \tag{7}$$

Note that the proofs of all the examples above are involved in the computation of the inverses of the exponents $\frac{1}{(e'-e)}, \frac{1}{(c-1)}$ and $\frac{1}{(e-1)}$ respectively, which requires the knowledge about the order of the group. However, since the order of $\mathbb{QR}_N^+$ group is unknown these protocols can not be trivially moved to $\mathbb{QR}_N^+$ group.

1.3 Our Contributions

The crux of the problem is that the inversion computation in the exponent is difficult in $\mathbb{QR}_N^+$ group with unknown order. To tackle this problem, we introduce a new proof approach called k-th power, which does not require the inversion computations in exponents. The k-th power approach extends the key technical tool developed by Hofheinz et al. which reduces the factoring problem to SDH problem in $\mathbb{QR}_N^+$ (Theorem 2,[6]).

The k-th power proof approach is almost generic, i.e., applying k-th power technique to many, if not all, existing authenticated Diffie-Hellman key exchange protocols under gap assumption immediately produces protocols under factoring assumption if they work over $\mathbb{QR}_N^+$.

As one application of k-th power approach, we show that FS protocol [4], in which k is a constant, is provably secure in eCK model under factoring assumption if it works over $\mathbb{QR}_N^+$.

Our technique also applies to other protocols, e.g., UP[15],KFU_1[9],HMQV[10] and its variants FHMQV,SMQV [13, 14], in which k is a non-constant, but at the cost of degrading a factor in the reduction.

1.4 Related Work

Cash et al. [3] introduced the "Twin Diffie-Hellman (TDH)" technique and showed that SDH assumption is implied by the standard CDH assumption. However, to apply TDH technique [12, 7–9], they have to modify the protocol at a cost of doubling computational overhead. In comparison, our technique directly yield a security proof under factoring assumption without modifying the protocol.

Boyd et al. [2] and Fujioka et al. [5] proposed authenticated key exchange protocols which can be instantiated under factoring assumption. However, their protocols are based on the generic CCA encryption scheme and thus clearly less efficient.

2 Preliminaries

Let the value κ be the security parameter. We write $[N] = \{1, ..., N\}$. For group elements g, X, we denote by $\log_g X$ the discrete logarithm of X to the base g.

2.1 Factoring Assumption

A prime number P is called safe prime, if $P = 2p + 1$ for some prime number p. We assume that $N = PQ$ where P, Q are safe prime numbers, and thus N is a blum integer number. Let $\text{RSAgen}(1^\kappa)$ be an algorithm that generates such elements (N, P, Q). For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$,

$$Pr[\mathcal{A}(N) = \{P, Q\}] \leq \epsilon(\kappa)$$

where $(N, P, Q) \leftarrow_R \text{RSAgen}(1^\kappa)$, and $\epsilon(\kappa)$ is negligible.

2.2 Signed Quadratic Residues

The set $\mathbb{QR}_N$ of quadratic residues modulo N is defined as $\mathbb{QR}_N := \{x \in \mathbb{Z}_N^* : \exists y \text{ and } x = y^2\}$. Since $\mathbb{Z}_N^* \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_{pq}$, $\mathbb{QR}_N \in \mathbb{Z}_N^*$ is cyclic group of order pq. By $\mathbb{J}_N$, we denote the subgroup of $\mathbb{Z}_N^*$ with Jacobi symbol 1.

For $x \in \mathbb{Z}_N$ we define $|x|$ as the absolute of x, where x is represented as a signed integer in the set $\{-(N-1)/2, ..., (N-1)/2\}$. We define the group of signed quadratic residues as $\mathbb{QR}_N^+ = \{|x| : x \in \mathbb{QR}_N\}$, where the group operation is defined by $g \circ h = |gh \bmod N|$. As all the computations will take place in $\mathbb{QR}_N^+$, we will omit the absolute values and simply write xy or $x \cdot y$ for $x \circ y$. The following facts have been noted in [6].

1. $(\mathbb{QR}_N^+, \circ)$ is a group of order $\phi(n)/4$, where $\phi(n)$ is Euler's totient function.
2. $\mathbb{QR}_N^+ = \mathbb{J}_N^+$ where $\mathbb{J}_N^+ = \mathbb{J}_N \bigcap [(N-1)/2]$. Thus, given only N the membership in $\mathbb{QR}_N^+$ is efficiently recognizable.
3. If $\mathbb{QR}_N$ is cyclic, so is $\mathbb{QR}_N^+$.

2.3 Strong Diffie-Hellman (SDH) Assumption

Let $\mathbb{G} = \langle g \rangle$ be the cyclic group whose order is not necessarily known. Define $\mathrm{CDH}(X, Y) := g^{(\log_g X)(\log_g Y)}$ where $X, Y \in \mathbb{G}$. For our purpose, we consider group $\mathbb{QR}_N^+$. For any probabilistic polynomial time algorithm $\mathcal{A}$,

$$Pr[\mathcal{A}^{\mathrm{DDH}_{g,X}(\cdot,\cdot)}(N, g, X, Y) = \mathrm{CDH}(X, Y)] \leq \epsilon(\kappa)$$

where $X, Y \leftarrow \mathbb{QR}_N^+$, and $\epsilon(\kappa)$ is negligible. $\mathrm{DDH}_{g,X}(\cdot,\cdot)$ denotes that $\mathcal{A}$ has oracle access to DDH, which given a two-tuples $(\hat{Y}, \hat{Z})$ in $\mathbb{QR}_N^+$, outputs 1 if $\hat{Y}^{\log_g X} = \hat{Z}$ and 0 otherwise.

The following theorem (Theorem 2,[6]) shows that in the $\mathbb{QR}_N^+$ group, the SDH problem is implied by the factoring assumption.

Theorem 1 ([6]). *If the factoring assumption holds then the strong DH assumption holds. In particular, for every SDH problem adversary $\mathcal{A}$, there exists a factoring adversary $\mathcal{B}$ with roughly the same complexity as $\mathcal{A}$.*

3 Our New Techniques and Applications

Before introducing our new idea, we recall the key technical tool developed by Hofheinz et al., which proved that in $\mathbb{QR}_N^+$ group SDH problem is implied by the factoring assumption (Theorem 2,[6]). We sketch the main idea as follows. For our convenience, the notations are slightly changed.

Factoring$\longrightarrow$SDH. A factoring algorithm $\mathcal{B}$, which uses a SDH adversary $\mathcal{A}$, is constructed as follows. $\mathcal{B}$ chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, $\mathcal{B}$ chooses $x, b \in [N/4]$ and sets

$$g = h^2 \qquad X = hg^x \qquad B = hg^b$$

This implicitly defines $\log_g X = x + \frac{1}{2} \bmod \mathrm{ord}(\mathbb{QR}_N^+)$, and $\log_g B = b + \frac{1}{2} \bmod \mathrm{ord}(\mathbb{QR}_N^+)$.

$\mathcal{B}$ can implement the SDH oracle, i.e., answers $\mathcal{A}$'s oracle queries $\hat{X}, \hat{Z} \in \mathbb{QR}_N^+$ (the membership is efficiently recognizable) by checking $\hat{X}^{2b+1} \stackrel{?}{=} \hat{Z}^2$, which is equivalent to $\hat{X}^{\log_g B} \stackrel{?}{=} \hat{Z}$.

Finally, $\mathcal{A}$ output $Z = g^{(\log_g X)(\log_g B)} = g^{(x+1/2)(b+1/2)} = h^{2xb+x+b+1/2}$, from which $\mathcal{B}$ can extract $v = h^{1/2}$ with the knowledge about x, b. Now, with two non-trivial different square roots u, v of h, $\mathcal{B}$ can factor N by $\gcd(u - v, N)$ (or $\gcd(u + v, N)$).

Our New Technique. However, the situation for authenticated key exchange protocols is different. Since the secret values $Z_i (i = 1, 2, ..m)$ of key derivation function $\mathrm{H}(\cdot)$, where m is the number of the secret values, are usually a combination of the static/ephemeral public keys of two parties, and thus are not usually of the form $CDH(X, B)$ from which $v = h^{1/2}$ is extracted. In particular,

it is hard to compute $CDH(X, B)$ from $Z_i (i = 1, 2, ..m)$ if the inversion computations in exponent are required because the order of $\mathbb{QR}_N^+$ group is unknown. For example, in order to compute $CDH(X, B)$ from Z_1, Z_2 the simulator of FS protocol requires the computation of the inverses $\frac{1}{(c-1)}$ in the exponent.

To tackle this problem, we extend Hofheinz et al.'s technique and introduce a new proof approach called k-th power. We sketch the main idea in the terminology of authenticated key exchange protocol. Assume that the Test session is $\Pi^{s^*}_{\hat{A},\hat{B}}$. Assume that the adversary queries the static private key of $\hat{A}$ and does not query the ephemeral private key of Test session.

$\mathcal{B}$ chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, $\mathcal{B}$ chooses $x, b \in [N/4]$, and an additional value $\boxed{\text{k}}$, which is related to the protocol (e.g. $k = c - 1$ for FS protocol), and sets

$$g = h^{2k} \qquad X = hg^x \qquad B = hg^b$$

This implicitly defines $\log_g X = x + \frac{1}{2k} \bmod \text{ord}(\mathbb{QR}_N^+)$, and $\log_g B = b + \frac{1}{2k} \bmod \text{ord}(\mathbb{QR}_N^+)$. $\mathcal{B}$ sets the outgoing message of Test session and the static public key of $\hat{B}$ to be X, B respectively.

Now, $\mathcal{B}$ can keep the consistency of the oracle queries against session $\Pi^s_{\hat{B},\hat{D}}$. Take FS protocol as an example: upon receipt of queries $(\hat{X}, \hat{Z}_i (i = 1, 2))$, $\mathcal{B}$ computes $\bar{Z}_1 = \hat{Z}_1/(\hat{X}D)^y$, $\bar{Z}_2 = \hat{Z}_2/(\hat{X}D^c)^y$ where $Y = g^y$ is maintained by $\mathcal{B}$, and hence

$$(\hat{X}D)^{\log_g B} = \bar{Z}_1 \Longleftrightarrow (\hat{X}D)^{2k \cdot \log_g B} = \bar{Z}_1^{2k} \Longleftrightarrow (\hat{X}D)^{(2kb+1)} = \bar{Z}_1^{2k}$$

$$(\hat{X}D^c)^{c \cdot \log_g B} = \bar{Z}_2 \Longleftrightarrow (\hat{X}D^c)^{2k \cdot c \cdot \log_g B} = \bar{Z}_2^{2k} \Longleftrightarrow (\hat{X}D^c)^{c \cdot (2kb+1)} = \bar{Z}_2^{2k}$$

Thus, $\mathcal{B}$ can check the correctness of the value $\hat{Z}_i (i = 1, 2)$ by checking whether $(\hat{X}D)^{(2kb+1)} \stackrel{?}{=} \bar{Z}_1^{2k}$ and $(\hat{X}D^c)^{c \cdot (2kb+1)} \stackrel{?}{=} \bar{Z}_2^{2k}$.

Finally, $\mathcal{A}$ output $Z_i (i = 1, 2, ..m)$ from which $\mathcal{B}$ can extract the value of the form $\bar{Z} = CDH(X, B)^k$, e.g., for FS protocol $\bar{Z} = CDH(X, B)^k = CDH(X, B)^{(c-1)}$. While it is difficult to the compute inversion operation $\frac{1}{k}$ in exponent, since

$$\bar{Z} = CDH(X, B)^k = g^{(\log_g X)(\log_g B)k} = g^{(x+1/2k)(b+1/2k)k} = h^{(2k \cdot xb + x + b + 1/2k)k}$$

$\mathcal{B}$ can extract $v = h^{(1/2k)k} = h^{1/2}$ with the knowledge about x, b. Thus, with two non-trivial different square roots u, v of h, $\mathcal{B}$ can factor N by $\gcd(u - v, N)$ (or $\gcd(u + v, N)$). No inversion computations in exponents are required throughout the proof.

Applications. The k-th power proof approach is almost generic, i.e., applying k-th power technique to many, if not all, existing authenticated Diffie-Hellman key exchange protocols under gap assumption immediately produces protocols under factoring assumption if they work over $\mathbb{QR}_N^+$.

Note that the value k varies with the protocols and should be set to be some value σ whose inversion operation $\frac{1}{\sigma}$ in exponent is required.

Applying k-th power technique to protocols with constant σ, e.g., in FS protocol $\sigma = (c-1)$, immediately produces protocols provably secure in eCK model under factoring assumption if they work over $\mathbb{QR}_N^+$.

For protocols with non-constant σ, e.g., UP[15] in which $\sigma = e - 1$, our technique also works but at the cost of degrading a factor in the reduction.

4 Authenticated Key Exchange Protocols Based on Factoring Assumption

In this section, we discuss the protocol with constant k and defer the discussion on the protocols with non-constant k to Section 5.

4.1 The Scheme with Constant k

FS protocol over $\mathbb{QR}_N^+$. A typical example with constant k is FS protocol, which is most efficient one (example 2, [4]) among FS protocol family. Setting $c = 2$ results in an efficient protocol with 3 exponentiations. FS protocol was originally described in a cyclic group of known prime order q and provably secure in eCK model under gap assumption. Here we present the protocol over $\mathbb{QR}_N^+$ in Fig. 1 which is provably secure in eCK model under the factoring assumption. Our proof technique also applies to other protocols of FS protocol family with the different choices of the value k.

Let the value κ be the security parameter. Let $N = PQ$ be a RSA modulus generated by RSAgen(1^κ). Let $\mathbb{QR}_N^+ = \langle g \rangle$ be a cyclic group of order pq. Let $H : \{0,1\}^* \to \{0,1\}^{l(\kappa)}$ be a hash function.

The party Alice($\hat{A}$)'s static private key is $a \in [N/4]$ and its static public key is $A = g^a \in \mathbb{QR}_N^+$. Similarly, the party Bob($\hat{B}$)'s static private key is $b \in [N/4]$ and its static public key is $B = g^a \in \mathbb{QR}_N^+$. We omit writing explicitly " mod N" for calculations modulo N.

4.2 Security

Theorem 2. *Suppose that the factoring assumption holds for RSAgen, H is a hash function modeled as random oracle, then the proposed protocol in Fig. 1 is a secure authenticated key exchange protocol in the eCK model described in Appendix A.*

Proof. The first condition of Definition 3 follows immediately from the correctness of our protocol. That is, if two parties complete matching sessions, then they compute the same key. The proof for second condition of Definition 3 consists of showing that the probability that the adversary distinguishes a real session key from a random string is not more than $\frac{1}{2}$ plus a negligible fraction. Since all exchanged information and identities are included in $\mathbf{H}(\cdot)$ which is modeled as a random oracle, the probability that a non-matching session has the same session key with the Test session is negligible. Thus, the only way that the adversary

$\hat{A}$	$\hat{B}$
$(A = g^a)$	$(B = g^b)$
$x \in [N/4], X = g^x$	$y \in [N/4], Y = g^y$
$\xrightarrow{X}$	
$\xleftarrow{Y}$	
$Z_1 = (YB)^{x+a}$	$Z_1 = (XA)^{y+b}$
$Z_2 = (YB^c)^{x+ac}$	$Z_2 = (XA^c)^{y+bc}$
$sk = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B})$	$sk = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B})$

Fig. 1. FS protocol over $\mathbb{QR}_N^+$

M succeeds is *Forging attack*, in which the adversary M computes the values Z_1, Z_2 itself and then queries $\mathbf{H}$ with $(Z_1, Z_2, X, Y, \hat{A}, \hat{B})$.

To show that the success probability of Forging attack is negligible, we will construct a factoring problem solver SIM that uses an adversary M who succeeds with non-negligible probability in the attack. Assume that there are n honest parties $\hat{U_1}, \hat{U_2}, ..., \hat{U_n}$, and at most m sessions are activated.

– **Input to SIM.** The input to SIM is a factoring problem instance $N = PQ$. The goal of SIM is to compute P or Q.

According to freshness definition, there are two complementary cases that the adversary chooses the Test session: Test session without a matching session and Test session with a matching session.

4.2.1 Test Session Has No Matching Session

It suffices to discuss the following two subcases that: the adversary issues *either* CASE 1: a StaticKeyReveal query on party $\hat{A}$ *or* CASE 2: EphemeralKeyReveal query on the Test session.

CASE 1: SIM chooses $i, j \in \{k|\hat{U_k}\}$, and $s^* \in [m]$. We denote $\hat{U_i}, \hat{U_j}$ by $\hat{A}, \hat{B}$ respectively. With these choices, SIM guesses that the adversary M will select the session $\Pi^{s^*}_{\hat{A},\hat{B}}$ as the Test session.

SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $b \in [N/4]$ and computes $g = h^{2k}, B = hg^b$, where k=c-1. This implicitly defines $\log_g B = b + \frac{1}{2k} \bmod \text{ord}(\mathbb{QR}_N^+)$. SIM sets the static public key B for $\hat{B}$, and random static key pairs for the remaining parties (including $\hat{A}$). SIM interacts with the adversary M as follows. Without loss of generality, we assume that $\hat{B}$ is the responder.

– $\mathbf{H}(Z_1, Z_2, X, Y, \hat{U_i}, \hat{U_j})$: SIM maintains an initially empty list H^{list} with entry of the form $(Z_1, Z_2, X, Y, \hat{U_i}, \hat{U_j}, h)$. SIM simulates the oracle in usual

way except for queries of the form $(Z_1, Z_2, X, Y, \hat{U}_i, \hat{U}_j)$ with $\hat{U}_i = \hat{D}, \hat{U}_j = \hat{B}$, i.e., we assume that $\hat{B}$ is the responder communicating with a peer $\hat{D}$. SIM responds to these queries in the following way:

- If $(Z_1, Z_2, X, Y, \hat{D}, \hat{B})$ is already there, then SIM responds with stored value h.
- Otherwise, if there are the entries of the form $(X, Y, \hat{D}, \hat{B}, *)$ in L^{list} (maintained in the Send query), SIM computes $\bar{Z}_1 = Z_1/(XD)^y, \bar{Z}_2 = Z_2/(XD^c)^y$, and hence

$$(XD)^{\log_g B} = \bar{Z}_1 \Longleftrightarrow (XD)^{2k \cdot \log_g B} = \bar{Z}_1{}^{2k} \Longleftrightarrow (XD)^{(2kb+1)} = \bar{Z}_1{}^{2k} \quad (8)$$

$$(XD^c)^{c \cdot \log_g B} = \bar{Z}_2 \Longleftrightarrow (XD^c)^{2k \cdot c \cdot \log_g B} = \bar{Z}_2{}^{2k} \Longleftrightarrow (XD^c)^{c \cdot (2kb+1)} = \bar{Z}_2{}^{2k} \quad (9)$$

  Thus, SIM can check the correctness of the value $Z_i (i = 1, 2)$ by checking whether $(XD)^{(2kb+1)} \stackrel{?}{=} \bar{Z}_1{}^{2k}$ and $(XD^c)^{c \cdot (2kb+1)} \stackrel{?}{=} \bar{Z}_2{}^{2k}$. If the equalities hold, it returns from L^{list} the stored value SK to the adversary M, stores the new tuple $(Z_1, Z_2, X, Y, \hat{D}, \hat{B}, SK)$ in H^{list}.
- Otherwise, SIM chooses h at random, sends it to the adversary M and stores the new tuple $(Z_1, Z_2, X, Y, \hat{D}, \hat{B}, h)$ in H^{list}.

– **StaticKeyReveal(U_i)**:
  - If $U_i = \hat{B}$, then SIM aborts.
  - Otherwise, SIM returns the corresponding static private key to the adversary M.
– **EstablishParty($\hat{U}_i$)**: The adversary can arbitrarily register a user on behalf of the party $\hat{U}_i$. This way, the adversary totally controls the party $\hat{U}_i$.
– **EphemeralKeyReveal($\Pi^s_{U_i,U_j}$)**:
  - If $\Pi^s_{U_i,U_j}$ is the Test session $\Pi^{s^*}_{\hat{A},\hat{B}}$, then simulator fails.
  - Otherwise, SIM returns the stored ephemeral private key to the adversary M.
– **Send($\Pi^s_{\hat{U}_i,\hat{U}_j}, m$)**: SIM maintains an initially empty list L^{list} with entries of the form $(X, Y, \hat{U}_i, \hat{U}_j, SK)$. SIM simulates the oracle in usual way except for Test session and the sessions of party $\hat{B}$. SIM responds to these queries in the following way:
  - If $\Pi^s_{\hat{U}_i,\hat{U}_j}$ is the Test session $\Pi^{s^*}_{\hat{A},\hat{B}}$, SIM chooses $x \in [N/4]$, returns $X^* = hg^x$ to the adversary M.
  - If $\hat{U}_i = \hat{B}$, SIM chooses $y \in [N/4]$ and returns $Y = g^y$ to the adversary M. (For convenience, we assume that $\hat{B}$ is the responder, and denote $\hat{U}_j$ by $\hat{D}$ and m by X.)
    * SIM looks in H^{list} for the entry of the form $(*, *, X, Y, \hat{D}, \hat{B}, *)$. If finds it, SIM computes $\bar{Z}_1 = Z_1/(XD)^y, \bar{Z}_2 = Z_2/(XD^c)^y$. Then, SIM can check the correctness of the value $Z_i (i = 1, 2)$ by checking whether $(XD)^{(2kb+1)} \stackrel{?}{=} \bar{Z}_1{}^{2k}$ and $(XD^c)^{c \cdot (2kb+1)} \stackrel{?}{=} \bar{Z}_2{}^{2k}$.
      · If the equality does not hold, SIM chooses SK randomly and stores the new tuple $(X, Y, \hat{D}, \hat{B}, SK)$ in L^{list}.

· Otherwise, SIM stores the new tuple $(X, Y, \hat{D}, \hat{B}, h)$ in L^{list} where the value h is the last element from H^{list}.

∗ Otherwise (no such entries exist), SIM chooses SK at random and stores the new tuple $(X, Y, \hat{D}, \hat{B}, SK)$ in L^{list}.

– **SessionKeyReveal**($\Pi^s_{\hat{U}_i,\hat{U}_j}$):
  - If $\Pi^s_{\hat{U}_i,\hat{U}_j}$ is the Test session $\Pi^{s^*}_{\hat{A},\hat{B}}$, then simulator fails.
  - Otherwise, SIM returns the stored value SK in L^{list} to the adversary M.

– **Test**($\Pi^s_{\hat{U}_i,\hat{U}_j}$):
  - If $\Pi^s_{\hat{U}_i,\hat{U}_j}$ is not the Test session $\Pi^{s^*}_{\hat{A},\hat{B}}$, SIM aborts.
  - Otherwise, SIM randomly chooses ζ and returns it to the adversary M.

Finally, if the adversary M provides a correct guess at $Z_1 = (Y^*B)^{\log_g X^* + \log_g A}$, $Z_2 = (Y^*B^c)^{\log_g X^* + c\log_g A}$ where X^* is the outgoing message of Test session, and Y^* is the incoming message from the adversary, SIM proceeds with following steps:

$$\bar{Z}_1 = Z_1/(Y^*B)^{\log_g A} \tag{10}$$

$$\bar{Z}_2 = Z_2/(Y^*B^c)^{c\log_g A} \tag{11}$$

$$Z = \frac{\bar{Z}_2}{\bar{Z}_1} = \frac{(Y^*B^c)^{\log_g X^*}}{(Y^*B)^{\log_g X^*}} = B^{(c-1)\log_g X^*} = B^{k\log_g X^*} \tag{12}$$

Hence,

$$Z = B^{k\log_g X^*} = g^{k(\log_g B)(\log_g X^*)} = g^{k(b+\frac{1}{2k})(x+\frac{1}{2k})} = h^{k(2kbx+b+x+\frac{1}{2k})} \tag{13}$$

From (13), SIM can extract $v = h^{(k/2k)} = h^{1/2}$ with the knowledge about x, b. Thus, with two non-trivial different square roots u, v of h, SIM can factor N by $\gcd(u - v, N)$ (or $\gcd(u + v, N)$).

CASE 2: The setup of SIM is identical to that of CASE 1 except that SIM chooses $a, b \in [N/4]$ and computes $g = h^{2k}$, $A = hg^a$, $B = hg^b$, where k=c(c-1). SIM sets the static public key A, B for $\hat{A}$ and $\hat{B}$ respectively, and random static key pairs for the remaining $n - 2$ parties.

– $\mathbf{H}(Z, X, Y, \hat{U}_i, \hat{U}_j)$: SIM simulates the oracle in usual way except for queries with $\hat{U}_i = \hat{A}$ or $\hat{B}$, or $\hat{U}_j = \hat{A}$ or $\hat{B}$. For these queries the action of SIM is similar to that of CASE 1 for queries of the form $(Z, X, Y, \hat{U}_i, \hat{U}_j)$ with $\hat{U}_i = \hat{D}$, $\hat{U}_j = \hat{B}$.

– **StaticKeyReveal**(U_i):
  - If $U_i = \hat{B}$ (or $\hat{A}$), then SIM aborts.
  - Otherwise, SIM returns the corresponding static private key to the adversary M.

- **EstablishParty**(U_i): The action of SIM is identical to that of CASE 1.
- **EphemeralKeyReveal**($\Pi^s_{U_i,U_j}$): SIM returns the stored ephemeral private key to the adversary M (including that of the Test session).
- **SessionKeyReveal**($\Pi^s_{U_i,U_j}$): The action of SIM is identical to that of CASE 1.
- **Send**($\Pi^s_{U_i,U_j}, m$):
  - If $\Pi^s_{U_i,U_j}$ is the Test session $\Pi^{s^*}_{\hat{A},\hat{B}}$, SIM chooses $x \in [N/4]$, returns $X^* = g^x$ instead of $X^* = hg^x$. Otherwise,
    - If $U_i = \hat{B}$ (or $\hat{A}$), the simulation is similar to that of CASE 1 for party $\hat{B}$.
    - Otherwise, ($U_i \neq \hat{B}$ and $U_i \neq \hat{A}$), since SIM knows the static private key it follows the protocol specification.
- **Test**($\Pi^s_{U_i,U_j}$): The action of SIM is identical to that of CASE 1.

Finally, if the adversary M provides a correct guess at $Z_1 = (Y^*B)^{\log_g X^* + \log_g A}$, $Z_2 = (Y^*B^c)^{\log_g X^* + c\log_g A}$ where X^* is the outgoing message of Test session, Y^* is the incoming message from the adversary, SIM proceeds with following steps:

$$\bar{Z}_1 = Z_1/(Y^*B)^{\log_g X^*} \tag{14}$$

$$\bar{Z}_2 = Z_2/(Y^*B^c)^{\log_g X^*} \tag{15}$$

$$Z = \frac{\bar{Z}_2}{\bar{Z}_1{}^c} = \frac{(Y^*B^c)^{c\log_g A}}{(Y^*B)^{c\log_g A}} = B^{c(c-1)\log_g A} = B^{k\log_g A} \tag{16}$$

Hence,

$$Z = B^{k\log_g A} = g^{k(\log_g A)(\log_g B)} = g^{k(a+\frac{1}{2k})(b+\frac{1}{2k})} = h^{k(2kab+a+b+\frac{1}{2k})} \tag{17}$$

From (17), SIM can extract $v = h^{(k/2k)} = h^{1/2}$ with the knowledge about a, b. Thus, with two non-trivial different square roots u, v of h, SIM can factor N by $\gcd(u - v, N)$ (or $\gcd(u + v, N)$).

4.2.2 Test Session Has a Matching Session

It suffices to consider the following four subcases.

CASE 3: The adversary issues the EphemeralKeyReveal queries on both the Test session and its matching session.

- The action of SIM is identical to that of CASE 2. However, as the value Y^* is from the matching session maintained by the simulator SIM, a more concise proof strategy is as following.
- SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}^*_N)^+ \backslash \mathbb{QR}^+_N$ and sets $h = u^2$. Then, SIM chooses $a, b \in [N/4]$ and computes $g = h^{2k}, A = hg^a, B = hg^b$, where $\boxed{\text{k=1}}$. Then, SIM sets the static public keys of party $\hat{A}$ and $\hat{B}$ to be A, B respectively.

– The simulation is identical to CASE 2. However, the reduction is more direct as SIM knows $\log_g Y^*$. From value $Z_1 = (Y^*B)^{\log_g X^* + \log_g A}$, with the knowledge about $\log_g X^*$ and $\log_g Y^*$, SIM directly derives $Z = CDH(A,B) = g^{(\log_g A)(\log_g B)} = g^{(a+\frac{1}{2k})(b+\frac{1}{2k})} = h^{2k(ab+\frac{1}{2k}a+\frac{1}{2k}b+\frac{1}{4k^2})} = h^{(2ab+a+b+\frac{1}{2})}$.
Then, SIM can extract $v = h^{1/2}$ with the knowledge about a, b. Thus, with two non-trivial different square roots u, v of h, SIM can factor N by $\gcd(u-v, N)$ (or $\gcd(u+v, N)$).

CASE 4: The adversary issues the StaticKeyReveal queries on both the party $\hat{A}$ and its peer $\hat{B}$.

– SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $x, y \in [N/4]$ and computes $g = h^{2k}, X^* = hg^x, Y^* = hg^y$, where $\boxed{\text{k=1}}$. Then, SIM sets the ephemeral public keys of Test session and matching session to be X^*, Y^* respectively.
– The simulation is simple as SIM knows all the static private keys. The reduction is as follows. From value $Z_1 = (Y^*B)^{\log_g X^* + \log_g A}$, with the knowledge about $\log_g A$ and $\log_g B$, SIM directly derives $Z = CDH(X^*, Y^*) = g^{(\log_g X^*)(\log_g Y^*)} = g^{(x+\frac{1}{2k})(y+\frac{1}{2k})} = h^{2k(xy+\frac{1}{2k}x+\frac{1}{2k}y+\frac{1}{4k^2})} = h^{(2xy+x+y+\frac{1}{2})}$.
Then, SIM can extract $v = h^{1/2}$ with the knowledge about x, y. Thus, with two non-trivial different square roots u, v of h, SIM can factor N by $\gcd(u-v, N)$ (or $\gcd(u+v, N)$).

CASE 5: The adversary issues the StaticKeyReveal query on the party $\hat{A}$ and the EphemeralKeyReveal query on the matching session.

– The action of SIM is identical to that of CASE 1. The more concise proof strategy is as following.
– SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $a, b \in [N/4]$ and computes $g = h^{2k}, X^* = hg^x, B = hg^b$, where $\boxed{\text{k=1}}$. Then, SIM sets the ephemeral public key of Test session and the static public keys of $\hat{B}$ to be X^* and B respectively.
– The simulation is identical to CASE 1. However, the reduction is more direct as SIM knows $\log_g Y^*$. From value $Z_1 = (Y^*B)^{\log_g X^* + \log_g A}$, with the knowledge about $\log_g A$ and $\log_g Y^*$, SIM directly derives $Z = CDH(X^*, B) = g^{(\log_g X^*)(\log_g B)} = g^{(x+\frac{1}{2k})(b+\frac{1}{2k})} = h^{2k(xb+\frac{1}{2k}x+\frac{1}{2k}b+\frac{1}{4k^2})} = h^{(2xb+x+b+\frac{1}{2})}$
Then, SIM can extract $v = h^{1/2}$ with the knowledge about x, b. Thus, with two non-trivial different square roots u, v of h, SIM can factor N by $\gcd(u-v, N)$ (or $\gcd(u+v, N)$).

CASE 6: The adversary issues the EphemeralKeyReveal query on the Test session and the StaticKeyReveal query on the party $\hat{B}$.

– This case is symmetric to CASE 5, and omitted.

Together with all the subcases CASE 1-CASE 6, the success probability of SIM is

$$Pr[SIM] \geq \max\{\max_{i=1,2,3,5,6}\{\frac{1}{mn^2}p_i\}, \frac{1}{m^2}p_4\} \tag{18}$$

where p_i is the probability of the event that the cases occurs and the adversary M succeeds in this case. If there is an adversary M who succeeds with non-negligible probability in any cases above, we can solve the factoring problem. This completes the proof of Theorem 2.

5 The Schemes with Non-constant k

UP Protocol over $\mathbb{QR}_N^+$. An example with non-constant k is UP protocol [16] which was originally described in a cyclic group of known prime order q and provably secure under gap assumption. Here we show that UP protocol is provably secure under the factoring assumption if it works over $\mathbb{QR}_N^+$.

$\hat{A}$	$\hat{B}$
$(A = g^a)$	$(B = g^b)$
$x \in [N/4], X = g^x$	$y \in [N/4], Y = g^y$
$\xrightarrow{X}$	
$\xleftarrow{Y}$	
$Z_1 = (YB^e)^{x+a}$	$Z_1 = (XA)^{y+be}$
$Z_2 = (YB)^{x+ad}$	$Z_2 = (XA^d)^{y+b}$
where $e = h(Y)$, $d = h(X)$	where $e = h(Y)$, $d = h(X)$
$sk = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B})$	$sk = H(Z_1, Z_2, X, Y, \hat{A}, \hat{B})$

Fig. 2. UP protocol over $\mathbb{QR}_N^+$

Theorem 3. *Suppose that the factoring assumption holds for RSAgen, h, H are hash functions modeled as random oracles, then UP protocol over $\mathbb{QR}_N^+$ (Fig. 2) is a secure authenticated key exchange protocol in the eCK model described in Appendix A.*

Sketch of proof. The proof is similar to that of section 4.1 with a difference that in the setup SIM has to guess the value k as it is not a constant, which results in a loss of a factor. In the following, we provide a rough discussion on the setting of value k, and the more details will be given in the full version. Assume that Test session is $\Pi^{s^*}_{\hat{A},\hat{B}}$. We take into account the following two cases in which the setting of the value k is different.

CASE 1: SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $k, b \in [N/4]$ and computes $g = h^{2k}, X^* = hg^x, B = hg^b$. SIM sets the ephemeral public key of Test session and the static public key of $\hat{B}$ to be X^* and B respectively.

In the interaction with the adversary M, SIM answers the query $\mathrm{h}(Y^*)$ with value $k+1$ where Y^* is the incoming message of Test session. This implicitly sets $k = e - 1$ where $e = \mathrm{h}(Y^*)$.

CASE 2: SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $k_1, k_2, b \in [N/4]$ and computes $k = k_1 k_2, g = h^{2k}, A = hg^a, B = hg^b$. SIM sets the static public keys of $\hat{A}$ and $\hat{B}$ to be A and B respectively.

In the interaction with the adversary M, SIM sets $\mathrm{h}(X^*) = k_1$ and $\mathrm{h}(Y^*) = k_2 + 1$ where X^* is the outgoing message of Test session, and Y^* is the incoming message of Test session. This implicitly sets $k = (e-1)d$ where $e = \mathrm{h}(Y^*)$ and $d = h(X^*)$.

HMQV Protocol over $\mathbb{QR}_N^+$. HMQV protocol [10] is another typical example with a non-constant k which was originally described in a cyclic group of known prime order q and provably secure under gap assumption. Here we show that HMQV protocol is provably secure under the factoring assumption if it works over $\mathbb{QR}_N^+$.

$\hat{A}$	$\hat{B}$
$(A = g^a)$	$(B = g^b)$
$x \in [N/4], X = g^x$	$y \in [N/4], Y = g^y$
$\xrightarrow{X}$	
$\xleftarrow{Y}$	
$Z = (YB^e)^{x+ad}$	$Z = (XA^d)^{y+be}$
where $e = h(Y, \hat{A})$, $d = h(X, \hat{B})$	where $e = h(Y, \hat{A})$, $d = h(X, \hat{B})$
$sk = H(Z, X, Y, \hat{A}, \hat{B})$	$sk = H(Z, X, Y, \hat{A}, \hat{B})$

Fig. 3. HMQV protocol over $\mathbb{QR}_N^+$

Theorem 4. *Suppose that the factoring assumption holds for RSAgen, h, H are hash functions modeled as random oracles, then HMQV protocol over $\mathbb{QR}_N^+$ (Fig. 3) is a secure authenticated key exchange protocol in the eCK model described in Appendix A.*

Sketch of proof. We provide a rough discussion on the setting of value k, and the more details will be given in the full version. The proof strategy also applies to the variants of HMQV, e.g., CMQV,FMQV and SMQV. Assume that Test session is $\Pi^{s^*}_{\hat{A},\hat{B}}$. We take into account the following two cases in which the setting of the value k is different.

CASE 1: SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $k_1, k_2, x, b \in [N/4]$ and computes $k = k_1 - k_2, g = h^{2k}, X^* = hg^x, B = hg^b$. SIM sets the ephemeral public key of Test session and the static public key of $\hat{B}$ to be X^* and B respectively.

In the interaction with the adversary M, SIM answers the query $\mathrm{h}(Y^*, \hat{A})$ with value k_1 where Y^* is the incoming message of Test session. In the repeat experiment, SIM sets $\mathrm{h}(Y^*, \hat{A})$ to be k_2. This implicitly sets $k = e - e'$ where e, e' are two different response values of $\mathrm{h}(Y^*, \hat{A})$ in Forking lemma.

CASE 2: SIM chooses uniformly $u \leftarrow_R (\mathbb{Z}_N^*)^+ \backslash \mathbb{QR}_N^+$ and sets $h = u^2$. Then, SIM chooses $k_1, k_{21}, k_{22}, b \in [N/4]$ and computes $k = (k_{21} - k_{22})k_1, g = h^{2k}, A = hg^a, B = hg^b$. SIM sets the static public keys of $\hat{A}$ and $\hat{B}$ to be A and B respectively.

In the interaction with the adversary M, SIM sets $\mathrm{h}(X^*, \hat{B}) = k_1$ and $\mathrm{h}(Y^*, \hat{A}) = k_{21}$ where X^* is the outgoing message of Test session, and Y^* is the incoming message of Test session. In the repeat experiment, SIM sets $\mathrm{h}(Y^*, \hat{A})$ to be k_{22}. This implicitly sets $k = (e - e')d$ where $d = \mathrm{h}(X^*, \hat{B})$ and e, e' are two different response values of $\mathrm{h}(Y^*, \hat{A})$ in Forking lemma.

Acknowledgments. The author would like to thank the anonymous reviewers for their valuable comments. This work was supported in part by Zhejiang Provincial Natural Science Foundation of China under Grant No. Y1110157, and Science Foundation of Zhejiang Sci-Tech University under Grant No. 1007827-Y.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Boyd, C., Cliff, Y., Gonzalez Nieto, J., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (2008)
3. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
4. Fujioka, A., Suzuki, K.: Designing efficient authenticated key exchange resilient to leakage of ephemeral secret keys. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 121–141. Springer, Heidelberg (2011)
5. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (2012)
6. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009)
7. Huang, H., Cao, Z.: Strongly secure authenticated key exchange protocol based on computational Diffie-Hellman problem. In: Proceedings of Inscrypt 2008, pp. 65–77. Science Press of China (2009), `http://eprint.iacr.org/2008/500`

8. Huang, H., Cao, Z.: An ID-based authenticated key exchange protocol based on bilinear Diffie-Hellman problem. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS, pp. 333–342. ACM (2009)
9. Kim, M., Fujioka, A., Ustaoğlu, B.: Strongly secure authenticated key exchange without NAXOS' approach. In: Takagi, T., Mambo, M. (eds.) IWSEC 2009. LNCS, vol. 5824, pp. 174–191. Springer, Heidelberg (2009)
10. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
11. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
12. Lee, J., Park, J.H.: Authenticated key exchange secure under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/344 (2008), `http://eprint.iacr.org/`
13. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A secure and efficient authenticated diffie–hellman protocol. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 83–98. Springer, Heidelberg (2010)
14. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010)
15. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography 46(3), 329–342 (2008)
16. Ustaoglu, B.: Comparing sessionstatereveal and ephemeralkeyreveal for Diffie-Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer, Heidelberg (2009)

A. Security Model

In this section, we review the eCK security model for authenticated key exchange protocols. For the details of the original eCK model, see [11, 15].

Participants. We model the protocol participants as a finite set $\mathcal{U}$ with each $U_i \in \mathcal{U}$ being a probabilistic polynomial time (PPT) Turing machine, which may execute a polynomial number of protocol instances in parallel. $\Pi^s_{U_i,U_j}(i,j \in N)$ denotes s-th instance of participant U_i with peer U_j.

Adversary Model. The adversary M is modeled as a PPT Turing machine and has full control of the communication network and may eavesdrop, delay, replay, alter and insert messages at will. We model the adversary's capability by providing it with oracle queries.

- **EphemeralKeyReveal**($\Pi^s_{U_i,U_j}$) The adversary obtains the ephemeral private key of $\Pi^s_{U_i,U_j}$.
- **SessionKeyReveal**($\Pi^s_{U_i,U_j}$) The adversary obtains the session key for a session s of U_i, provided that the session holds a session key.
- **StaticKeyReveal**(U_i) The adversary obtains the static private key of U_i.

- **EstablishParty(U_i)** The adversary can arbitrarily register a user on behalf of the party U_i. This way, the adversary totally controls the party U_i. If a party is registered by the adversary, then it is called *dishonest* (or *malicious*). Otherwise, it is called *honest*.
- **Send($\Pi^s_{U_i,U_j}, m$)** The adversary sends the message m to the session $\Pi^s_{U_i,U_j}$ and gets a response.
- **Test($\Pi^s_{U_i,U_j}$)** Only one query of this form is allowed for the adversary. A random bit $\hat{b}$ is chosen, if $\hat{b} = 0$ then the real session key is returned; otherwise, an uniformly chosen random value ζ is returned.

Definition 1 (Matching Session). *Let $\Pi^s_{U_i,U_j}$ be a completed session with identifier $(U_i, U_j, out, in, role)$, where U_i is the owner of the session, U_j is the peer, and out is U_i's outgoing message, in is U_j's outgoing message, and role is the U_i's role in the session (initiator or responder). The session $\Pi^t_{U_j,U_i}$ is called the* matching session *of $\Pi^s_{U_i,U_j}$, if the identifier of $\Pi^t_{U_j,U_i}$ is $(U_j, U_i, \overline{out}, \overline{in}, \overline{role})$, where $out = \overline{in}, in = \overline{out}, role \neq \overline{role}$.*

Definition 2 (Freshness of eCK model). *Let instance $\Pi^s_{U_i,U_j}$ be a completed session, which was executed by an honest party U_i with another honest party U_j. We define $\Pi^s_{U_i,U_j}$ to be* fresh *if none of the following three conditions hold:*

- *The adversary M reveals the session key of $\Pi^s_{U_i,U_j}$ or of its matching session (if latter exists).*
- *U_j is engaged in session $\Pi^t_{U_j,U_i}$ matching to $\Pi^s_{U_i,U_j}$ and M issues either:*
  - *both **StaticKeyReveal**(U_i) and **EphemeralKeyReveal**($\Pi^s_{U_i,U_j}$) queries; or*
  - *both **StaticKeyReveal**(U_j) and **EphemeralKeyReveal**($\Pi^t_{U_j,U_i}$) queries.*
- *No sessions matching to $\Pi^s_{U_i,U_j}$ exist and M issues either:*
  - *both **StaticKeyReveal**(U_i) and **EphemeralKeyReveal**($\Pi^s_{U_i,U_j}$) queries; or*
  - ***StaticKeyReveal**(U_j) query.*

As a function of the security parameter κ, the advantage of the PPT adversary M in attacking protocol Σ is defined as $Adv^{AKE}_{M,\Sigma}(\kappa) \stackrel{def}{=} |Pr[b = \hat{b}] - \frac{1}{2}|$, where $Pr[b = \hat{b}]$ is the probability that the adversary queries **Test** oracle to a *fresh* session, outputs a bit b which is equal to the bit $\hat{b}$ of **Test** oracle.

Definition 3 (AKE Security). *An authenticated key exchange protocol Σ is said to be AKE-secure if following two conditions hold*

1. *If two parties complete the matching sessions, they compute the same session key.*
2. *For any PPT adversary M, the probability $Adv^{AKE}_{M,\Sigma}(\kappa)$ is negligible.*