# Authenticated Key Exchange Protocol with Remote Randomness

John C. W. Chan

City University of Hong Kong
johnchan713@gmail.com

**Abstract.** A conventional Authenticated Key Exchange (AKE) protocol consumes fresh random coins from the local random source. However, recent studies of bad randomness expose the vulnerability of some AKE protocols under small subgroup attacks when the random coins are manipulated or being repeated. It is important to ensure the bad randomness of one random source will not affect the security of the AKE protocol as a whole. Thus, we introduce the notion of remote randomness by introducing additional ephemeral keys generated by a fresh remote random source in the AKE protocol. In this paper, we argue that because of the thrive of cloud computing, it encourages high speed internal data transfer within server clusters or virtual machines, including entropy pool data used in our remote randomness AKE protocols. We present a remote randomness modification to the HMQV protocol to demonstrate its resilience under the manipulation of local random sources. We then provide a new security model with the consideration of remote randomness and show that the modified AKE protocol is secure under our new model.

**Keywords:** Cryptography, Remote Randomness, Remote Randomness Model, Remote Randomness Protocol, RHMQV

## 1 Introduction

An Authenticated Key Exchange (AKE) protocol is a collection of cryptographic algorithms which enables two parties to authenticate each other and establishing a common session key to encrypt the messages transmitted over the insecure public network. To ensure the security of an AKE protocol, the random coins consumed by pseudo-random generators (PRNGs) must be fresh in order to guarantee high entropy [10] of generated parameters used in an AKE session. Due to efficiency, AKE protocol consumes pre-generated and locally stored random coins in the form of an entropy pool, which contains collections of randomized measurements like CPU temperature, or fan speeds. Once the entropy pool is compromised under adversarial manipulations [1], leakage [19,20], or reversions [17], the bad randomness will affect all randomized parameters of a party in an AKE session.

It is a danger if an AKE protocol relies on merely the local random source. An adversarial manipulation by controlling the values of random coins [1] will result in predictable ephemeral key values since the algorithm of some widely used PRNGs, such as ANSI X9.17 PRNG is deterministic and public. Before the public keys are generated, leakage of randomness or using weak randomness sources [20] will result in insufficient entropy of those keys generated by PRNGs that they are not indistinguishable from truly random strings. The repeat of random coins by adversarial resets of instances [1] or reversion of virtual machines from stored states [17] will result in the reuse of ephemeral key [7] and lead to compromise of the session key even if no information regarding the ephemeral key is exposed to an adversary. In this paper, we argue that because of the thrive of cloud computing and the increasingly available infrastructures of server clusters and virtual machines, the drawbacks brought by bad randomness can possibly be relieved.

The cloud computing relies on efficient server architectures. Typically, cloud servers are arranged in the form of distributed systems where data can be efficiently transmitted within the server clusters. One efficient arrangement is called the mesh architecture that multiple servers are arranged as a web like topology interconnected with pipes of optic fibers. A cloud can support multiple virtual machines (VMs) run on the same server through the monitor and control of hypervisor like Xen hypervisor [21]. Those VMs are called guest systems which allows heterogeneous operating systems and can be launched on randomly chosen cloud servers, or even being migrated to other servers during the running of an instance. The job of adversaries on cloud will be harder in tracing internal transfers and identifying guest systems involved in a particular AKE session.

Since bad randomness is sufficient to contaminate the local entropy pool as a whole, in avoidance to "put all eggs in one basket", we introduce the notion of Remote Randomness that an AKE protocol not only consumes random coins from the local random source, but also randomness from a remote machine. As long as one of the source is fresh, we can show that our RHMQV protocol is resilient to bad randomness. The RRAKE protocol consumes random coins of a randomly chosen servers or VMs. To achieve this, we require a high speed internal data transfer like cloud since the entropy pool data transferred need to be consumed directly by an AKE protocol and stateless on local machine to avoid both random sources to be repeated by the reversion of VM in [17].

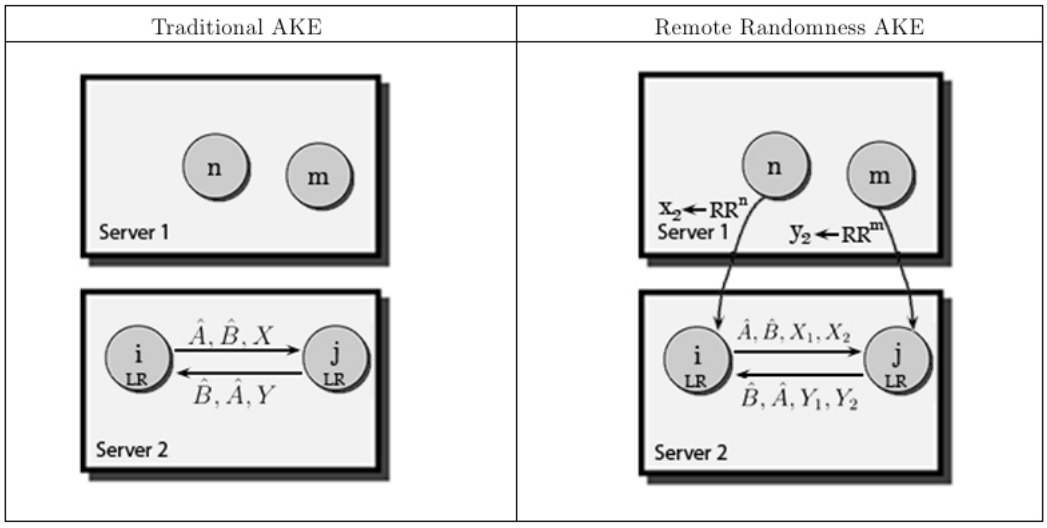


Fig. 1. Contrast between traditional AKE and remote randomness AKE

This paper was the first formal study on security model and AKE protocol which utilize two randomness sources in each party. We propose Remote Randomness Model (RRM) based on the Bellare-Rogaway (BR) [8], Canetti-Krawczyk (CK) [9], and reset models [1] to capture both the adoption of remote random source, and the situation of local bad randomness. We design the RRAKE protocol by introducing the concept of Local Ephemeral Key (LEK) and Remote Ephemeral Key (REK) in an AKE protocol. LEK is generated by the PRNG consuming the local random coins, which is the same as the ephemeral key in the original AKE protocol. REK is generated as the same way as LEK but using a remote random source. Both keys are to be transmitted in an AKE session. We shows a remote randomness modification to HMQV protocol [3], which is known as the RHMQV protocol with an improved hashed challenge-response signatures (HCR) can resist to all types of adversarial manipulations to local random coins.

The paper will be arranged as follows. For section 2, we have a closer look to the previous studies of bad randomness and their solutions in contrast to our remote randomness solutions. Next, we will show the detrimental effect to rely on local random source by proposing a reset attack in section 3. Our new remote randomness security model (RRM) and the proof of remote randomness security will be introduced in section 4. In section 5, we illustrate the security of remote randomness HMQV (RHMQV) protocol under our new model. This paper is concluded in section 6.

## 2 Related Works

Cryptographic primitives under the manipulation or leakage of randomness are extensively studied in [1, 7, 10, 17, 19, 20]. Yang et al. [1] proposed security models to capture bad randomness in AKE protocols when random coins values are specified (reset-1 model), or repeated (reset-2 model) by adversary manipulations. Bellare et al. [10] introduces the hedged public key encryption that is secure with indistinguishabiliy under chosen distribution attack (IND-CDA) when employing weak random sources. The study of leakage of randomness in [19, 20] demonstrates the risk of exposure of partial random coins values to the adversary result in low entropy of the key values. For the reuse of randomness, Yilek [17] shows the revert state functionality of VMs can lead to the repeat of randomness in an AKE session, Menezes and Ustaoglu [7] shows the insecurity of HMQV protocol under the reuse of ephemeral keys.

All the above studies assume that cryptographic primitives, including those employed by an AKE protocol, relies only on the local random source. Yilek [17] mentioned the efficiency of some primitives can be improved by reusing some of the randomness, but there are no formal security models to capture the scenario that a party is reusing part of its randomness in an AKE protocol. The solution of pseudo-random function (PRF) as a strong randomness extractor (SRE) towards reset attack in [1] are no more efficient than using a hashed challenge-response signature when fresh remote random coins can be directly consumed by the PRNG. The reversion of states result in reuse of ephemeral keys [7, 17] are not sufficient to render RRAKE insecure since the remote random coins are stateless and the generation of session key is not solely depends on the reused ephemeral keys, we will show this in section 5. Recently, studies on resettable cryptographic primitives like the resettable Zero-Knowledge proof [25] and resettable Identifiation protocol [23] used in smart cards, which are also based only on the local random source. Our work is different from [23, 25] that we examine the security on general AKE protocols in secure communication on messages rather than specifically designed protocols for verification, or used in the proof of knowledge.

A related concept of including two randomness in a primitive is the public randomness [22] that the input of public-key cryptosystems includes original random coins with a bit string from a public source, which is accessible by every parties in the system including the adversaries. However, the security of public randomness is assessed by $(t, \delta)$ concrete approach, which is different from the indistinguishability approach or simulation approach [8, 9, 11] used to assess the security of an AKE protocol, also efficient random oracles are not available in the primitive in [22]. Moreover, the public randomness is referred to a centralized random source whereas the remote source is a distributed random source that random coins are chosen randomly among servers or VMs with independent entropy pools. Bad randomness will affect all parties in the system for a centralized random source but not in a distributed source.

The HMQV protocol by Krawczyk [3] is a Diffie-Hellman AKE protocol with hashed challenge-response signatures (HCR). Menezes [4] pointed out the vulnerability of HMQV under the small subgroup attack when ephemeral key is reused with the omission of G-tests for key validation [5]. The small subgroup attack can be realized as a reset attack on VMs [1,17] which is effective towards several widely used AKE protocols like ISO protocol and SIGMA protocol [1] including the Identity-based AKE (IBAKE) protocols [12,15,24]. In this paper, we show an modification to HMQV protocol, which is known as RHMQV protocol, can resist to such attack when the local random coin is reused.

The Remote Randomness model (RRM) is based on the complexity-theoretic approach of AKE security, which is due to Bellare-Rogaway (BR) model [8] and Canetti-Krawczyk (CK) model [9]. BR-model simulates Key Exchange in a multi-party system that any party can initiate an instance to activate communication with another party, adversaries may engage in targeted sessions to perform cryptographic attacks, which is similar to the activities of guest systems on cloud. Moreover, the CK-model introduces strong adversaries which can obtain secrets in AKE session by merely issuing queries, it provides security proofs by simulation to AKE protocols like HMQV which cannot be easily assessed under the indistinguishability approach. Besides, the resets models [1] which is based on [9], captures the adversarial manipulations of random coins during the initiation stage of instances, but only one random coin is fed to an particular instance. The RRM is the modification of reset models with two random coins feed to an instance, but the reduction between the two models is not possible since the inputs of the two models are different.

## 3 Reset attacks on AKE

The reset attack is due to the small subgroup attack by Menezes and Ustaoglu [7] under the reuse of ephemeral keys and omission of key validation. Here we propose reset attacks on Hölbl et al.'s protocols [24], an Identification-based AKE protocol and shows the danger of using one local random source in AKE protocols. The reset attack uses queries in the reset-2 model by Yang et al. [1] which captures the repeat of random coins in AKE.

**Key Validation** The group membership check on the public key $pk_i \in [2, p-1]$ and $pk_i^q = 1$. Since the selection of modulo for the public keys is for randomly chosen p and q, given an even integer n such that $p = 1+nq$, then $(p-1)/q$ contains some smooth divisors greater than $q$ with sizes smaller than $2^{40}$. The collision of key choices with those divisors enables small subgroup attack by brute-force testing of exponents. The validation is an expensive operation that are omitted in some AKE protocols like HMQV for efficiency.

**Hölbl et al.'s protocol1** The key generation center (KGC) shares a large prime $p$, primitive root $g \in_R \mathbb{Z}_p^*$ and a hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$, a master secret key $s \in_R \mathbb{Z}_p^*$ that $y_s = g^s$ is the master public key. The KGC publishes the tuple $\{g, H, p, y_s\}$.
The Key Generation Center (KGC) first calculates $I_i = H(ID_i)$ in which $ID_i$ is the identity of party i. KGC then choose $k_i \in_R \mathbb{Z}_p^*$ and calculates public key $u_i = g^{k_i}$, The key pair $(k_i, u_i)$ is sent to the parties. The static private key is given by $v_i = I_i k_i + s u_i$.

Fig. 1. Hölbl et al.'s protocol1

| A | | B |
|---|---|---|
| $r_A \in_R \mathbb{Z}_p^*$ | | |
| | $u_A, T_A, ID_A$ | |
| $T_A = g^{r_A}$ | $\longrightarrow$ | $r_B \in_R \mathbb{Z}_p^*$ |
| | $u_B, T_B, ID_B$ | |
| $I_B = H(ID_B)$ | $\longleftarrow$ | $T_B = g^{r_B}$ |
| $X_A = u_B^{I_B} y_s^{u_B} = g^{v_B}$ | | $I_A = H(ID_A)$ |
| $K_{AB} = (X_A T_B)^{(v_A + r_A)}$ | | $X_B = u_A^{I_A} y_s^{u_A} = g^{v_A}$ |
| | | $K_{AB} = (X_B T_A)^{(v_B + r_B)}$ |

Both of the parties agreed on the same session key $K_{AB} = g^{(v_A+r_A)(v_B+r_B)}$.The ephemeral keys $r_A, r_B$ are generated form the entropy pools of party A and party B, while the key of one party can be repeated through a reset, the state-reveal query enables adversaries to obtain another ephemeral key value in the state. Without key validation, it enables the testing of private key $k_B$ by the following small subgroup attack.

Reset Attack on Hölbl et al.'s protocol1

The Hölbl et al.'s protocol1 runs on a group G of order q which is a subgroup of $\mathbb{Z}_p^*$ that $T = (p-1)/q$ which is greater than $q$, has some relatively prime factors such that $T = t_1 \cdot t_2 \cdot t_3 \ldots t_n$ and $\forall i, |t_i| < 2^{40}$.

1. The adversary $C$ corrupts party $A$ and obtain $A$'s private key $v_A = I_A k_A + s u_A$.
2. $C$ initiates new instance of $A$, selects $\gamma \in Z_p^*$ of order $t_1$ and sends the tuple $(u_A, T_A, ID_A)$ to $B$.
3. $C$ issues State-reveal query to obtain $B$'s ephemeral private key $r_B$.
4. $A$ receives the tuple $(u_B, T_B, ID_B)$ and calculates $I_B = H(ID)_B, K_{AB} = \beta^{(I_B k_B + s u_B + r_B)}$ where $\beta = g^{(I_A k_A + s u_A + r_A)}$.
5. The adversary issues Reveal query and obtain the session key $K_{AB} = g^{(I_A k_A + s u_A + r_A)(I_B k_B + s u_B + r_B)} = \beta^{(I_B k_B + s u_B + r_B)}$.
6. The adversary computes $K' = \beta^{(I_B c_1 + s g^{c_1} + r_B)}$ for $c_1 = 1, 2, 3, \ldots$ until $K' = K$ that $d_1 = k_B \mod t_1$.
7. The adversary resets party $A$ and repeats procedure 1 to 6 to obtain $d_i = k_B \mod t_i$ for $i = 1, 2, 3, \ldots$ and recovers $d = k_B \mod q$ with Chinese Remainder Theorem.

The acquisition of private key enables adversary to impersonate as party B in the subsequent sessions, which breaks the perfect forward secrecy (PFS). The reset attack on Hölbl et al.'s protocol2 is similar to the above except that the session key is calculated as $K_{AB} = g^{(k_A + sH(ID_A, u_A) + r_A)(k_B + sH(ID_B, u_B) + r_B)}$ where $u_i = g^{k_i}$, the private key $k_B$ can still be tested in the hash function $H(ID_B, g^{k_B})$. Another impersonation attack is available in [15] in which adversaries do not engage in the session.
In Hölbl protocol1, all long-lived parameters in the above protocol will be compromised by adversaries by corruption $(k_A, ID_A)$, messages received $(ID_B)$ and the output from the small subgroup attack $(k_B)$. In the strong corruption model with state-reveal query, the receiver's ephemeral key $r_B$ is exposed to the adversary, and the security of the whole protocol depend on one ephemeral key $r_A$. It exposes the insecurity of some widely used AKE protocol like HMQV [7], ISO [1], Unified Model and KEA which rely on ephemeral keys generated by local source of randomness.

## 4 Remote Randomness Model

We present Remote Randomness Model (RRM) which captures an AKE protocol utilizing both local and remote random sources. Let $S_U = \{U_1, U_2, \ldots, U_n\}$ be the set of parties in the system and each VM instance is indicated by a unique instance number $i \in I_U$ that $I_U$ is the set of VM instances initiated by party $U$, $I_*$ represents the set of instance numbers in the system such that $I_U \subseteq I_*$. The pair of parties in communication is indicated by the partner identifier $\mathsf{pid} = [ID_i, ID_j]$ and an AKE session is identified by a unique binary string of session identifier $s = [i, j, t]$ where $i, j$ are instance numbers with initiator goes first, $t$ is the time stamp of the session. It is required that in a matching AKE session, the partner identifier and the session identifier of the two communicating parties are the same. The system allows external parties to register to the system including benign parties and adversaries except that adversaries can issue the set of queries $\{\mathsf{Corrupt}, \mathsf{StateReveal}, \mathsf{SSKReveal}\}$ to reveal secrets in an AKE session. The security of an AKE protocol lies on whether an adversary participated in the security game can successfully output the session key without the help of adversarial queries and repeat of random coins in the test session. The simulation game is defined in the following nine procedures.

$\underline{\mathsf{Initialize}(k)}$
$(pk_*, sk_*) \leftarrow_{\$} \mathsf{SKG}(1^k)$
$Q_* \leftarrow 0; S_s \leftarrow \phi; S_U \leftarrow \bigcup_{p=1}^{n} U_p$
$b \leftarrow_{\$} \{0,1\}; I_* \leftarrow \phi$
ret $pk_*$

$\underline{\mathsf{Register}(U, pk)}$
If $(U \in S_U) \vee (\{pk_U\}_{U \in S_U})$ then ret $\perp$
$S_U \leftarrow \{U\} \cup S_U$
ret 1

$\underline{\mathsf{StartInstance}(U, i, rp_i)}$
If $(U \notin S_U) \vee (i \in I_*)$ then ret $\perp$
$I_U \leftarrow \{i\} \cup I_U; LR_U^i \leftarrow rp_i; St_U^i \leftarrow (LR_U^i, ready)$
$s_{i*} \leftarrow \perp; \mathsf{term}_{s_{i*}} \leftarrow 0; \mathsf{pid} \leftarrow \perp$
$\mathsf{acc}_{s_{i*}} \leftarrow 0; ssk_{s_{i*}} \leftarrow \perp$
ret 1

$\underline{\mathsf{Send}(U, i, M_{\text{in}})}$
If $(U \notin S_U) \vee (i \notin I_U) \vee (\mathsf{term}_s)$ then ret $\perp$
$RR_U^i \leftarrow \{rp_j\}_{j \in_R I_*, i \neq j}$
$(M_{\text{out}}, \mathsf{pid}, s, \mathsf{acc}_s, \mathsf{term}_s, ssk, St_U^i)$
$\leftarrow P(1^k, U, pk_U, sk_U, St_U^i, RR_U^i, M_{\text{in}})$
If $(\mathsf{acc})$ then $acc_s \leftarrow 1; ssk_s \leftarrow ssk$
ret $M_{\text{out}}, \mathsf{pid}, s, \mathsf{acc}_s, \mathsf{term}_s$

$\underline{\mathsf{Corrupt}(U)}$
If $(U \notin S_U)$ then ret $\perp$
$Q_{\mathsf{Corrupt}} \leftarrow 1$
ret $sk_U$

$\underline{\mathsf{StateReveal}(U, i)}$
If $(U \notin S_u) \vee (i \notin I_U)$ then ret $\perp$
$Q_{\mathsf{StateReveal}} \leftarrow 1$
ret $St_U^i$

$\underline{\mathsf{SSKReveal}(i, j, t)}$
If $(\mathsf{not\ acc}_s) \vee (i, j \notin I_*) \vee (\mathsf{term}_s)$ then ret $\perp$
$Q_{\mathsf{SSKReveal}} \leftarrow 1$
ret $ssk_s$

$\underline{\mathsf{Test}(V', j')}$
If $(V' \notin S_U) \vee (j' \in I_*) \vee (\mathsf{not\ acc}_s)$ then ret $\perp$
If $(b = 0)$ then
$N \leftarrow_{\$} \{0,1\}^{\ell(k)}$
ret $N$
Else ret $\{ssk\}_{s=[i',j',t]}$

$\underline{\mathsf{Finalize}(b')}$
$\mathsf{pid} \leftarrow [U', V']$
If $(U', V' \notin S_U)$ then ret $\perp$
If $(Q_{\mathsf{Corrupt}} \vee Q_{\mathsf{StateReveal}} \vee Q_{\mathsf{SSKReveal}})$ then ret $\perp$
If $(\exists i \neq i' \ni LR_U^i = LR_U^{i'}) \wedge$
$(\exists i \neq i' \ni RR_U^i = RR_U^{i'})$ then ret $\perp$
If $(\exists j \neq j' \ni LR_U^j = LR_U^{j'}) \wedge$
$(\exists j \neq j' \ni RR_U^j = RR_U^{j'})$ then ret $\perp$
ret $(b = b')$

**Fig. 2.** Security Game RRAKE

The details of the procedures are explained as follows.

**Initialize**$(k)$ The procedure to initialize the system. The randomized key generation algorithm $\mathsf{SKG}$ generates long-lived public keys and long-lived private keys $(pk, sk)$ for all parties in probabilistic polynomial time with length extension parameter $k$. The binary array $Q_* = [Q_{\mathsf{Corrupt}}, Q_{\mathsf{StateReveal}}, Q_{\mathsf{SSKreveal}}]$ which indicates the usage of the three adversarial queries is reset to zero. All session identifiers in set $S_s$ and instance numbers in set $I_*$ are cleared. All parties are initialized in the set $S_U$ and a bit $b$ is tossed for the use in the test query. Long-lived public keys will then delegated to all parties in the system.

**Register**$(U, pk)$ The procedure to add party $U$ with public key $pk$ into the system. A membership check is given to indicate whether the tuples $(U, pk)$ already exists in the system, the procedure will be aborted if there is a collision. **StartInstance**$(U, i, rp_i)$ The procedure to start a VM instance with random source $rp_i$. The instance number $i$ is added to the system after checking its uniqueness. The random source $rp_i$ will be extracted as the local random coin $LR_U^i$ and stored in the state $St_U^i$ of the VM when the coin is ready. Reset all parameters including partner identifier, session identifier, session key $(ssk)$ and decision bits of accept and terminate session $(\mathsf{acc}, \mathsf{term})$ in all previous sessions for party $i$ in which $s_{i*} = [i, *, *]$.

**Send**$(U, i, M_{in})$ The procedure to execute an AKE protocol. After the membership checking and confirm that the AKE session is not terminated, a remote random coin $RR_U^i$ is prepared by extracting the remote random source $rp_j$ from a randomly chosen instance $j$ in the system. The protocol then executed as a function $P(1^k, U, pk_U, sk_U, St_U^i, RR_U^i, M_{\text{in}})$ including protocol messages inputted $(M_{\text{in}})$, which depends on the AKE protocol used. The output of the function is $(M_{\text{out}}, \mathsf{pid}, s, \mathsf{acc}_s, \mathsf{term}_s, ssk, St_U^i)$ with $(M_{\text{out}})$ as the outputted protocol message. Note that the remote random coins is prepared at the time of protocol execution and consumes the random coins directly without storing in the state.

**Corrupt**$(U)$ This query exposes the private key $(sk)$ of party $U$ to the adversary. After the corruption, the adversary gains overall control over party $U$ in an AKE session including the active participation in the protocol as party $U$, and the possession of all internal states and intermediate parameters at party $U$ that Perfect Forward Secrecy is not guaranteed. The bit $Q_{\mathsf{Corrupt}} = 1$ indicates that the corrupt query is already issued.

**StateReveal**$(U, i)$ This query exposes the state of a VM instance to an adversary. It may include the local random coins, or the ephemeral key value of the specified party.

**SSKReveal**$(i, j, t)$ This query exposes the session key of a particular session to the adversary given that the specified session is accepted and not terminated by the parties.

**Test**$(V', j')$ The procedure to test the ability of the adversary in obtaining the session key given the challenge tuple $(V', j')$ as the targeted instance to attack. It is required that the targeted session is already accepted and this procedure can be called only once per game. If the bit $b$ flipped in the initialize procedure is 0, then a random string with identical length to the session key will be returned, otherwise the system will return the session key which belongs to the session between adversary and the targeted party $[i', j', t]$.

**Finalize**$(b')$ After the adversaries completed the test session, the game is finalized by checking whether the following conditions are true: 1.Targeted party $V'$ is a member in the system, 2. No adversarial queries are issued during the test session, 3. The local and remote random coins are not both repeated, that the adversary is considered successfully compromised the test session in this round of security game.

The Remote Randomness Model allows all forms of adversarial manipulations on randomness. The specification of value or repeat of random coins is to reuse the random source $rp_i$ whereas the leakage of the local random coin value is through StateReveal query. When the game is repeated to an infinite times, if the coin $b$ tossed in the Initialize phrase is zero, it enables the game to output a random string instead of the real session key. Then we come to the following definition.

**Definition 1** *Let $\Pi$ be an Remote Randomness AKE Protocol and $A$ be a Probabilistic Polynomial Time adversary, the advantage of $A$ against AKE-R in the RRAKE security game is given by*

$$\mathbf{Adv}^{\mathsf{RRAKE}}_{\Pi,A}(k) = \Pr[\mathsf{RRAKE}_{\Pi,A}(k) = 1] - \frac{1}{2} \tag{1}$$

with security parameter $k$, the advantage of adversary $A$ in the RRM is negligible.

The RHMQV adopts a new form of signature, the dual randomness exponential challenge response signatures (DR-XCR). The signature is an extension of Schnorr scheme which consists of three independent ephemeral keys in a Triangular Ephemeral Keys (TEK) arrangement. We first prove the security of the DR-XCR. Then we shows that the RHMQV is secure in the RRM.

**Definition 2** *(DR-XCR Signature Scheme). Let $\hat{B}$ be the signer with long-lived public key $B = g^b \in G$. A verifier $\hat{A}$ provides $\hat{B}$ a message $m$ and challenge $X$ and $\hat{B}$ signed and returns the tuple $(Y_1, Y_2, \text{DXSIG}(Y_1, Y_2, m, X))$ where $Y_1 = g^{y_1}$ is LEK and $Y_2 = g^{y_2}$ is an REK, $\hat{A}$ accept the tuple $(Y_1, Y_2, \sigma_B)$ if $\sigma_B = X^{y_1+eb}$ where $e = \bar{H}(Y_2, X, m)$ with $\bar{H}(.)$ an $\ell$-bit hash function.*

Similar to XCR scheme, the DR-XCR scheme is resistant to the replay attack by the inclusion of the identity of communicating parties as the message. We now define the security of the DR-XCR Signature by defining a Forger $\mathcal{F}$ in the forgery game 1.

**Definition 3** *(Security of DR-XCR). We say the DR-XCR signature scheme is secure if no polynomial time forger $\mathcal{F}$ can win the forgery game 1 with non-negligible probability.*

**Forgery Game 1.** The forger $\mathcal{F}$ is given the tuple $B, X' \in_R \mathcal{G}$ and polynomial times access to a signing oracle $\hat{B}$ with public key $B = g^b$. Upon a query $(X, m)$ the $\hat{B}$ outputs $(Y_1, Y_2, \text{DXSIG}(Y_1, Y_2, m, X))$ where $Y_1, Y_2$ are fresh values chosen by $\hat{B}$. Finally, $\mathcal{F}$ outputs a tuple $(Y_1', Y_2', m', \sigma')$. We said $\mathcal{F}$ is *successful forgery* if the followings are true:

1. $(Y_1', Y_2', \sigma')$ is a valid pair of signature from the challenge $(m', X')$.
2. $(Y_1', Y_2', \sigma')$ was not previously appear as the oracle responses of $\hat{B}$.

Here we require $(Y_1', Y_2')$ are both different from the previous queries. The probability of $\mathcal{F}$ can forge a signature is still negligible in this case.

**Proposition 1** *The DR-XCR signature is secure under CDH assumption and Random Oracle Model.*

*Proof.* Given a *successful forger* $\mathcal{F}$ we then construct a CDH solver $\mathcal{C}$ as follows:
**CDH Solver $\mathcal{C}$.** $\mathcal{C}$ input $(B = g^b, X')$ for signer $\hat{B}$ and provides $\mathcal{F}$ with random coins. $\mathcal{C}$ then simulate the hash query $\bar{H}$ by random answers during the run. Upon queries by $\mathcal{F}$ on $(X, m)$, $\mathcal{C}$ answer the query for $\hat{B}$ by choosing $s \in_R Z_q, e \in_R \{0,1\}^\ell$, set $Y_1 = g^s B^{-e}$. $\mathcal{C}$ set $\bar{H}(Y_2, X, m) = e$ if the result of query is outputted in previous queries, otherwise $C$ aborts the run. $\mathcal{C}$ answers $\mathcal{F}$ with the tuple $(Y_1, Y_2, X^s)$.

When $\mathcal{F}$ halts, $\mathcal{C}$ checks if the following conditions are satisfied. If not, $\mathcal{C}$ aborts the run.

1. $\mathcal{F}$ has outputted a tuple $(Y_1', Y_2', m', \sigma')$, $Y_1', Y_2' \in \mathcal{G}$.
2. $\bar{H}(Y_2', X', m')$ was queried from the oracle $\bar{H}$.
3. $(Y_1', Y_2', \sigma')$ was not used in any output of signature by $\hat{B}$.

**Repeat.** $\mathcal{C}$ runs the experiment for $\mathcal{F}$ the second time with the same input. However, the answers from $\mathcal{C}$ before $\bar{H}(Y_2', X', m')$ is the same as the first run, but a fresh random value $e' \in_R \{0,1\}^\ell$ is outputted by $\mathcal{C}$ on the query of $\bar{H}(Y_2', X', m')$ and subsequent queries.
**Output.** $\mathcal{C}$ checks if the output $(Y_1', Y_2', m', \sigma'')$ satisfy condition 1, 2, and 3 with hash value $e' \neq e$, then $\mathcal{C}$ output a guess $(\sigma'/\sigma'')^{(e-e')^{-1}}$ for the CDH problem $CDH(B, X')$.

Using the Forking lemma [26], if a forger $\mathcal{F}$ can succeed in forging the signature with the tuple $(Y_1', Y_2', m', \sigma')$ with the hash $\hat{H}(Y_2', X', m')$, the forger can also succeed in forging the same signature with a different hash value. It can be seen that the above simulation is perfect except with a negligible probability $1/2^\ell$, $\mathcal{F}$ successfully forge a valid signature without query to the hash oracle. Therefore we have

$$\left(\frac{\sigma'}{\sigma''}\right)^{(e-e')^{-1}} = \left(\frac{(Y_1 B^e)^{x'}}{(Y_1 B^{e'})^{x'}}\right)^{(e-e')^{-1}} = B^{x'} \tag{2}$$

a non-negligible probability that $\mathcal{C}$ can calculate $CDH(B, X')$ from tuple $(B, X')$. □

When two DR-XCR signers interact with each other signing the challenge provide by the partner, finally they will form the same signature. This signature is a dual DR-XCR.

**Definition 4** *(DR-DCR Signature Scheme). Let $\hat{A}, \hat{B}$ be two signers with long-lived public keys $A = g^a, B = g^b \in G$. After the exchange of challenges and both parties obtain the tuple $(m_A, m_B, X_1, X_2, Y_1, Y_2)$ where $X_1 = g^{x_1}, Y_1 = g^{y_1}$ are LEKs and $X_2 = g^{x_2}$, $Y_2 = g^{y_2}$ are REKs, the signature outputted by the two parties is the tuple $(X_1, X_2, Y_1, Y_2, \text{DDSIG}(m_A, m_B, X_1, X_2, Y_1, Y_2))$ with $\text{DDSIG}(m_A, m_B, X_1, X_2, Y_1, Y_2)$ $= g^{(x_1+da)(y_1+eb)}$ where $d = \bar{H}(X_2, Y_1, m_B)$ and $e = \bar{H}(Y_2, X_1, m_A)$.*

It is trivial to see that $(Y_1 B^e)^{(x_1+da)} = (X_1 A^d)^{(y_1+eb)}$ and the signature of the two parties are the same. The security of the DR-DCR signature and the proof is similar to DR-XCR except that the signed message and challenge are different.

**Definition 5** *(Security of DR-DCR). We say the DR-DCR signature scheme is secure if no polynomial time forger $\mathcal{F}$ can win the forgery game 2 with non-negligible probability.*

**Forgery Game 2.** We modify the forgery game 1 as follows, the query to signer $\hat{B}$ become $(X_1, X_2, m_A, m_B)$ and the output of $\hat{B}$ becomes $(Y_1, Y_2, \text{DXSIG}(Y_1, Y_2, m_A, X_1 A^d)$ where $d = \bar{H}(X_2, Y_1, m_B)$. The rest of the game are the same.

**Proposition 2** *The DR-DCR signature is secure under CDH assumption and Random Oracle Model with the knowledge of a.*

*Proof.* The proof is similar to the security of DR-XCR but with signature and hash messages defined above in definition 4. Moreover, the guess from the CDH solver $\mathcal{C}$, $(\sigma'/\sigma'')^{(e-e')^{-1}} \neq B^{x_1} = CDH(B, X_1)$ since $A^d$ is included in the challenge in which $d = \bar{H}(X_2, Y_1, m_B)$. In the repeated run, the value of $d'$ may be different from $d$ that

the forger $\mathcal{F}$ chooses another message $m'_B$ when querying to the hash oracle. That we perform the following transformation to the guess of $\mathcal{C}$

$$\sigma'_0 = \frac{\sigma'}{(Y_1 B^e)^{da}} = \frac{(X_1 A^d)^{y_1+eb}}{(Y_1 B^e)^{da}} = \frac{(Y_1 B^e)^{x_1+da}}{(Y_1 B^e)^{da}} = (X_1)^{y_1+eb} \tag{3}$$

The transformation for $\sigma''_0$ is the same except with signature $\sigma''$ and using $e'$ instead of $e$. Note that the private key $a$ is needed in the transformation. Finally, we have

$$\left(\frac{\sigma'_0}{\sigma''_0}\right)^{(e-e')^{-1}} = \left(\frac{(X_1)^{y_1+eb}}{(X_1)^{y_1+e'b}}\right)^{(e-e')^{-1}} = \left(\frac{(Y_1 B^e)^{x_1}}{(Y_1 B^{e'})^{x_1}}\right)^{(e-e')^{-1}} = B^{x_1} \tag{4}$$

a non-negligible probability that $\mathcal{C}$ can output $CDH(B, X_1)$ from tuple $(B, X_1)$. □

**Proposition 3** *The RHMQV under CDH assumption and random oracle model is secure under the RRM.*

**Proof Idea**. We say two parties $\hat{A}$ and $\hat{B}$ engaged in a matching session if their session ID and party ID are the same. If an adversary $\mathcal{A}$ can successfully output the session key $s$ in a matching session with non-negligible probability larger than $1/2$, $\mathcal{A}$ can also success in the forgery of DR-DCR signature with non-negligible probability.

The methods that $\mathcal{A}$ can output the correct session key in the test session are (1) Direct guess with success probability $1/2^k$ where $k$ is the session key length, which is negligible. (2) Making two non-matching sessions output the same session key that the adversary can use either one in test session, this also happens in negligible probability. (3) The adversary use a forger to output a session key.

*Proof.* The proof of method (3) is as follows, suppose an adversary $\mathcal{A}$ can output a correct session key in RRAKE game with non-negligible probability, then we construct a CDH solver $\mathcal{C}$ to solve the CDH problem given the long-lived private key of party $\hat{A}$ is exposed to the adversary.
We first construct an adversary $\mathcal{A}_1$ who simulates the RRAKE Game for $\mathcal{A}$ as follows. $\mathcal{A}_1$ chooses a random session $s = [i', j', t']$ with $\mathsf{pid} = [U' = \hat{A}, V' = \hat{B}]$ that $\mathcal{A}_1$ targeted to output the session key in the test session. $\mathcal{A}_1$ simulates the game to $\mathcal{A}$ by answering all queries in the game on its own. If $\mathcal{A}$ cannot output a matching session $s = [i', j', t']$ in the test session, $\mathcal{A}_1$ aborts the game. In other cases, if $\mathcal{A}$ asks for private key of party $V'$ through the corruption query, $\mathcal{A}_1$ aborts the game. We denote $\mathsf{E}$ be the event that the adversary $\mathcal{A}$ outputs the same matching session as $\mathcal{A}_1$ with session key $s = [i', j', t']$ in the test session. For $i, j \in_R \{1, \ldots, n\}$ and time stamp $t \in_R \{1, \ldots, m\}$, the probability of a adversary $\mathcal{A}$ output a matching test session is $\Pr[\mathsf{E}] = (nm(n-1))^{-1}$.
Given an adversary $\mathcal{A}_1$ we then construct a CDH solver $\mathcal{C}$ with the same specification in proposition 2. We denote $\mathsf{F}$ the event that given $\mathcal{A}_1$ outputs the matching test session, $\mathcal{C}$ can output a guess $(\sigma'_0/\sigma''_0)^{(e-e')^{-1}} = CDH(B, X_1)$. Since this probability is negligible, combine all results, we have $(nm(n-1))^{-1}\Pr[\mathsf{F}]$, a negligible probability for $\mathcal{A}$ to output the correct session key in the RRM. □

## 5 Remote Randomness Protocol

We present the Remote Randomness Protocol modified from two-pass HMQV which utilizes two different random sources. $(x_1, y_1)$ are the local ephemeral private keys (LESK)

generated from local random source and $(x_2, y_2)$ are the remote ephemeral private keys (RESK) generated from a remote random source. Given $\bar{H}(.)$, a $\ell$ bit hash function where $\ell = (\lfloor log_2 q \rfloor + 1)/2$. To avoid confusion, $(pk_i, sk_i)$ denotes the long-lived public key and long-lived private key of party $i$.

Fig. 3. RHMQV protocol

| A | | B |
|---|---|---|
| $(pk_A, sk_A) = (g^a, a)$ | | $(pk_B, sk_B) = (g^b, b)$ |
| $x_1 \in_R \mathbb{Z}_q, X_1 \leftarrow g^{x_1}$ | | $y_1 \in_R \mathbb{Z}_q, Y_1 \leftarrow g^{y_1}$ |
| $x_2 \in_R \mathbb{Z}_q, X_2 \leftarrow g^{x_2}$ | | $y_2 \in_R \mathbb{Z}_q, Y_2 \leftarrow g^{y_2}$ |
| | $\hat{A}, \hat{B}, X_1, X_2$ $\longrightarrow$ | $d = \bar{H}(X_2, Y_1, \hat{B})$ |
| | | $e = \bar{H}(Y_2, X_1, \hat{A})$ |
| | $\hat{B}, \hat{A}, Y_1, Y_2$ $\longleftarrow$ | $\sigma_{\hat{B}} = (X_1 pk_A^d)^{(y_1+eb)}$ |
| $d = \bar{H}(X_2, Y_1, \hat{B})$ | | $K_{AB} = \sigma_{\hat{B}}$ |
| $e = \bar{H}(Y_2, X_1, \hat{A})$ | | |
| $\sigma_{\hat{A}} = (Y_1 pk_B^e)^{(x_1+da)}$ | | |
| $K_{AB} = \sigma_{\hat{A}}$ | | |

Both of the parties agree on the session key to the value of $g^{(x_1+da)(y_1+eb)}$. There are two exponent sets $s_1 = (x_1 + \bar{H}(X_2, Y_1, \hat{B})a)$ and $s_2 = (y_1 + \bar{H}(Y_2, X_1, \hat{A})b)$. When LEK is reused in party $A$ and the LEK of party B is exposed by StateReveal query, it will result in reused exponent of ephemeral keys $x_1$ to be fixed in $s_1$ and leaks the value of $Y_1$, but with $X_2$ a fresh random number results in $d$ to be floating after several resets. The same situation is true in exponent set $s_2$ in the reset attack that $Y_2$ is floating. Therefore, small subgroup attack is difficult against RHMQV protocol since the attack is not adequate to repeat or expose all ephemeral exponents to facilitate the value testing to long-lived keys. Compared to traditional HMQV protocol with $s_1 = (x + \bar{H}(X, B)a)$, the whole exponent set $s_1$ will be reused if party $A$ is being reset since $s_1$ is solely depend on the only ephemeral key $x$.

## 6 Conclusion and future work

In this paper, we introduce the notion of remote randomness in Authenticated Key Exchange (AKE). We proposed reset attacks to two Identification-based protocols to illustrate the harmful effect of using one randomness. We proposed a formal security model, the Remote Randomness Model (RRM) to capture remote randomness. We proposed RHMQV protocol, an AKE protocol with local and remote ephemeral keys. We showed the security of DR-XCR signature and its dual form, DR-DCR signature under the CDH assumption and Random Oracle model. We then showed the security of RHMQV protocol, which adopts the above signatures, under the Remote Randomness Model.
Our future work is to propose more AKE protocols with remote randomness.

# References


1. G. Yang, S. Duan, Duncan S. Wong, C. H. Tan, and H. Wang, "Authenticated Key Exchange under Bad Randomness", Financial Cryptography and Data Security, Lecture Notes in Computer Science, pp. 113-126, 2012.
2. H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol", Advances in Cryptology - CRYPTO '05, Lecture Notes in Computer Science, pp. 546-666, 2005.
3. H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol", Full version of [2], available at http://eprint.iacr.org/2005/176/.
4. A. Menezes, "Another Look at HMQV", Journal of Mathematical Cryptology, pp. 47-64, 2007.
5. A. Menezes, B. Ustaoglu, "On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols", INDOCRYPT 2006, LNCS 4329, Springer-Verlag, pp. 133-147, 2006.
6. S. Blake-Wilson, A. Menezes, "Authenticated Diffie-Hellman Key Agreement Protocols", SAC'98, LNCS 1556, pp. 339-361, 1999.
7. A. Menezes, B. Ustaoglu, "On Reusing Ephemeral Keys in Diffie-Hellman Key Agreement Protocols", International Journal of Applied Cryptography, Vol. 2,No. 2, pp. 154-158, 2010.
8. M. Bellare, P. Rogaway, "Entity Authentication and Key Distribution", Advances in Cryptology - CRPYPTO '93, Lecture Notes in Computer Science, Vol. 773, Springer-Verlag, 1994.
9. R. Canetti, H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", EUROCRYPT 2001, Springer-Verlag, pp. 453-474, No. 49, 2001.
10. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, S. Yilek, "Hedged Public-Key Encryption: How to Protect against Bad Randomness", ASIACRYPT 2009, LNCS 5912, pp. 232-249, 2009.
11. M. Bellare, R. Canetti, H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols", 30th STOC, pp. 419-428, 1998.
12. V. Kolesnikov, G. S. Sundaram, "IBAKE: Identity Based Authenticated Key Exchange Protocol", Cryptology ePrint Archive, Report 2011/612, http://eprint.iacr.org/, 2011.
13. A. P. Sarr, P. Elbaz-Vincent, J. Bajard, "A Secure and Efficient Authenticated Diffie-Hellman Protocol", CT-RSA 2010, LNCS 5985, pp. 41-56, 2010.
14. R. Canetti, H. Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol", CRYPTO 2002, LNCS 2442, pp. 143-161, 2002.
15. S. Zhang, Q. Cheng, X. Wang, "Impersonation Attack on Two Identity-Based Authenticated Key Exchange Protocols", 2010 WASE ICIE, 2010.
16. M. Bellare, R. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", ACM conference on Computer and communications security, pp. 62-73,November 1993.
17. S. Yilek, "Resettable Public-Key Encryption: How to Encrypt on a Virtual Machine", CT-RSA 2010, LNCS 5985, pp. 41-56, 2010.
18. R. Canetti, H. Krawczyk, "Universally Composable Notions of Key Exchange and Secure Channels", IACR's Eprint archive, http://eprint.iacr.org/2002, 2002.
19. H. Namiki, K. Tanaka, K. Yasunaga, "Randomness Leakage in the KEM/DEM Framework", ProvSec 2011, LNCS 6980, pp. 309-323, 2011.
20. M. Naor, G. Segev, "Public-Key Cryptosystems Resilient to Key Leakage.", CRYPTO 2009, LNCS 5677, pp. 18-35, Springer, Heidelberg 2009.
21. V. P. Chandu, K. Singh, "Performance Evaluation of a Xen-based Virtual Environment for High Performance Computing Systems", Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, pp. 475-481, Springer 2007.
22. A. Herzberg, M. Luby, "Public Randomness in Cryptography", proceedings to CRYPTO '92, Lecture Notes in Computer Science, pp. 421-432, Springer 1993.
23. M. Bellare, M. Fischlin, S. Goldwasser, S. Micali, "Identification Protocols Secure Against Reset Attacks", EUROCRYPT 2001, pp. 495-511, 2001.
24. M. Hölbl, T. Welzer, "Two improved two-party identity-based authenticated key agreement protocols", JCSI 2009, pp. 1056-1060, 2009.
25. V. Goyal, A. Sahai, "Resettably secure computation", EUROCRYPT 2009, pp. 54-71, 2009.
26. D. Pointcheval, J. Stern "Security Argument for Digital Signatures and Blind Signatures", Journal of Cryptology, Vol. 13, no. 3, pp. 361-396, Springer-Verlag, 2000.