Authenticated Key Exchange under Bad Randomness

Guomin Yang[1], Shanshan Duan[2], Duncan S. Wong[3], Chik How Tan[1], and Huaxiong Wang[4⋆]

[1] National University of Singapore
{tslyg,tsltch}@nus.edu.sg
[2] University of California San Diego
ssduan2009@gmail.com
[3] City University of Hong Kong
duncan@cityu.edu.hk
[4] Nanyang Technological University
hxwang@ntu.edu.sg

Abstract. We initiate the formal study on authenticated key exchange (AKE) under bad randomness. This could happen when (1) an adversary compromises the randomness source and hence directly controls the randomness of each AKE session; and (2) the randomness repeats in different AKE sessions due to reset attacks. We construct two formal security models, Reset-1 and Reset-2, to capture these two bad randomness situations respectively, and investigate the security of some widely used AKE protocols in these models by showing that they become insecure when the adversary is able to manipulate the randomness. On the positive side, we propose simple but generic methods to make AKE protocols secure in Reset-1 and Reset-2 models. The methods work in a modular way: first, we strengthen a widely used AKE protocol to achieve Reset-2 security, then we show how to transform any Reset-2 secure AKE protocol to a new one which also satisfies Reset-1 security.

Keywords: Authenticated Key Exchange, Resettable Cryptography, Bad Randomness

1 Introduction

An Authenticated Key Exchange (AKE) protocol consists of a tuple of randomized algorithms that enable two parties communicating over an insecure network to establish a common session key. These algorithms consume random coins which are typically generated by pseudo-random number generators (PRNGs). Practical PRNGs, such as ANSI X9.17 PRNG and FIPS 186 PRNG, can generate bit-strings which are computationally

⋆ The work of H. Wang is supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

indistinguishable from truly random strings provided that the seeds of the PRNGs are fresh and truly random [16]. In practice (e.g. OpenSSL), seeds are formed by collecting random data from an entropy pool, which can be created from a hardware source (e.g., sound/video input, disk drives, etc.) and/or a non-hardware source (e.g., the timing and content of events such as mouse movement, keystroke, network traffic, etc.) [19, 30].

In some practical situations however, the entropy pool could be controlled by an adversary and the seeds may no longer be fresh or truly random. For example, if an adversary has physical access to a hardware source and/or can control the events of a non-hardware source [19], the adversary may be able to manipulate the data in the entropy pool. Also, in some operating systems such as Linux, random data from the entropy pool may be pre-generated and stored in a buffer for later use. If the buffer is not well-protected, an adversary may be able to modify these pre-generated random data.

Adversarial reset of machines could make an AKE protocol reuse the same random coins in different sessions. It has been known as a real threat to some computing devices such as smart cards [12, 6]. Recently, Ristenpart and Yilek [34] showed that the adversarial reset is also a serious security threat to Virtual Machines (VMs). Generally, a system administrator can take snapshots of the current system state of a VM from time to time as regular backups. The VM can later be reverted back to a previous state using the snapshots. To perform an adversarial reset, an adversary can first make a system on a VM crash, e.g., via a Denial of Service (DoS) attack. Then when the system administrator reverts the VM back to a "good state", some random coins that have occurred before the machine being reset would be reused after the VM is reverted [34].

We categorize the threats above into "Reset-1" and "Reset-2" attacks, respectively. In Reset-1 attack, the adversary controls the random coins used by the AKE algorithms, while in Reset-2 attack, the adversary can reset a device to make algorithms reuse some random coins. The practical interests of these attacks bring a natural question: would the existing AKE protocols, especially those widely used ones, be still secure under these attacks?

In this paper, we conduct the first formal study on AKE under Reset-1 and Reset-2 attacks. Below are the three aspects to which we contribute.

Security Models. We propose two formal security models for Reset-1 and Reset-2 attacks respectively. We build our models based on the existing Bellare-Rogaway (BR) [7] and Canetti-Krawczyk (CK) [13] security models by providing the adversary additional capabilities. In the Reset-1

model, the adversary directly picks random coins for the AKE participants, while in the Reset-2 model, the adversary does not pick random coins directly but can reset a participant so that the same random coins will be used in multiple AKE sessions. In addition, to capture Kaliski's online *Unknown Key Share* (UKS) attacks [11], our models allow the adversary to register malicious users with public keys of its own choice.

In the Reset-1 model, since the adversary already controls the randomness, if the adversary also learns the long-lived key of either participant, it can trivially compute the session key. Hence the model cannot allow the adversary to corrupt the long-lived key of either participant, and *Forward Secrecy* (FS) cannot be captured. On the other side, the Reset-2 model does not have this restriction. This also indicates that the two models are incomparable and explains the reason why we need two models.

Security of Existing Protocols. Based on the two new models we defined, we show that some well-known AKE protocols (e.g. ISO [24, 13], SIGMA[1] [14, 27], JFK [2], SKEME [28], HMQV [26]), which are proven secure in their original security models will become insecure when the adversary is allowed to manipulate the randomness in the way defined in Reset-1 and/or Reset-2.

Designing New Protocols. We then present techniques to build AKE protocols that are secure in both Reset-1 and Reset-2 models. We present a generic way to efficiently transform a Reset-2 secure AKE protocol to a new one which is secure in both Reset-1 and Reset-2 models. Our idea is to generate good "internal" randomness within the protocol, we do this by applying a pseudo-random function (PRF) to the "raw" randomness at the very beginning of a protocol so that after this treatment, computationally "good" randomness are used in the remaining steps of the protocol. And we prove that this simple (but useful) idea indeed works. However, we remark that some additional requirement on the PRF is needed in order to make the transformed protocol still be Reset-2 secure.

Our transformation provides a "modular approach" to the construction of Reset-1 and Reset-2 secure AKE protocols: (1) we first build a Reset-2 secure AKE protocol Π; (2) then we apply the transformation to Π and get a new protocol Π' which maintains Reset-2 security and in addition to this, it also satisfies Reset-1 security. We illustrate this modular approach by proposing a new SIG-DH based AKE protocol which is transformed from the ISO protocol. The modifications we made are simple and efficient, and can be deployed easily to existing implementations of the protocol.

[1] SIGMA serves as the basis of the signature-based mode of IKE [23] and IKEv2 [25].

2 Related Work

Authenticated key exchange (AKE) protocols have been extensively studied within the cryptographic community. The first complexity-theoretic treatment of the security notion of AKE is due to Bellare and Rogaway [7]. Their model (referred to as the BR model) and its variants (e.g., [8, 9, 13]) then became the *de facto* standard for analyzing AKE protocols. In [13], Canetti and Krawczyk combined previous work and presented a new security model (referred to as the CK model). They showed that AKE protocols secure in their model can be composed with symmetric key encryption and authentication functions to provide provably secure communication channels. The CK model was used to demonstrate the security of many popular AKE protocols such as the ISO protocol [24, 13], the SIGMA protocol [14, 27], the HMQV protocol [26] and many more. Recently, LaMacchia et al. [29] extended the CK model to a new model (referred to as eCK model). In their model, the adversary is allowed to compromise either the long-live keys or the ephemeral keys (the latter are related to the randomness) of the participants of a protocol session. There have been many discussions on the strength of the two (CK and eCK) models [29, 32, 10]. In [10], Boyd et al. suggested that these two models are incomparable. In this paper, we follow the definitional approach of Canetti and Krawczyk, and we will see later that in fact our Reset-2 model can be considered as Resettable CK model.

None of the existing security models for AKE considers the bad randomness scenarios that we describe in this paper. Although for AKE protocols resilience to the leakage of ephemeral secret key has been studied by Krawczyk [26] and by LaMacchia et al. [29], their work is different from ours: firstly, ephemeral secret key is related but not equivalent to the randomness required by an AKE protocol, in particular, randomness may be required in other parts of the protocol; secondly, in the case of ephemeral secret key leakage, the adversary can only *passively* learn the ephemeral secret key, but not control its value. Another piece of work that is "somehow" related to ours is the work by Aiello et al. [2] in which the JFK (Just Fast Keying) AKE protocol was proposed and discussions about the reuse of Diffie-Hellman (DH) exponents in multiple JFK AKE sessions were made. However, this is different from the Reset-2 scenario we discussed earlier. The reuse of DH exponent is initiatively implemented by a participant of the protocol for reducing the number of costly modular exponentiation operations. But in a reset attack, all the components of the protocol use unfresh/used randomness. To see the difference more

clearly, if an AKE protocol (such as JFK) uses a randomized digital signature scheme (such as Digital Signature Standard - DSS [1]), then reusing the same randomness to sign different messages may allow an adversary to derive the secret signing key. However, merely reusing the DH exponent may not cause such a serious consequence.

Resettable Cryptography. Resettable security have been considered for other cryptographic protocols before, such as resettable Zero-Knowledge (rZK) proof [12] and resettable Identification (rID) protocols [6]. Recently, Goyal and Sahai [22] studied the problem of resettably secure two-party and multi-party computation for general functionalities, and in [35], Yilek studied resettably secure public key encryption. Although AKE can be considered as a two-party computation function, our work is different from that of Goyal and Sahai [22]. We focus on examining and enhancing the existing AKE protocols that have been widely used in the real practice.

Hedged Cryptography. Our paper is not the first paper to treat bad randomness for cryptographic operations. The Hedged Cryptography [4, 34] preprocesses randomness together with other inputs (messages, keys, etc.) of a cryptographic operation to provide (pseudo)randomness for the cryptographic operation. In particular, in [34], Ristenpart and Yilek presented the hedged RSA key transport and authenticated Diffie-Hellman key exchange protocols used in TLS without formal security models. Their heading technique is different from our treatment to the randomness presented in Sec. 5 and their hedged protocols cannot provide Reset-1 security.

3 Security Models and Definitions

3.1 AKE Protocol Descriptions

An Authenticated Key Exchange (AKE) protocol consists of two probabilistic polynomial time algorithms: the Long-Lived Key generation algorithm SKG and a protocol execution algorithm P. In this paper, we focus on the public key setting where the algorithm SKG returns a public key and a private key upon each invocation.

Protocol Participants. We initialize a nonempty set $\mathcal{U}$ of parties. Each party $U \in \mathcal{U}$ is named by a unique string, and that string has some fixed length. We use another set $\mathcal{MU}$ to denote malicious parties who are added into the system by an adversary after the initialization phase. Each malicious party $M \in \mathcal{MU}$ is also named by a distinct and fixed-

length string which has never been used to name another party inside the system.

Long-Lived Keys. Each party $U \in \mathcal{U}$ holds a public/private key pair $(\mathrm{pk}_U, \mathrm{sk}_U)$ that is generated according to the Long-Lived Key generation algorithm SKG. However, for each party $M \in \mathcal{MU}$, its public key pk_M can be set to any value except that pk_M has never been used as the public key of another party inside the system.

Instances. A party may run many instances concurrently. We denote instance i of party U by Π_U^i. At the time a new instance is created, a unique instance number within the party is chosen, a sequence of random coins are tossed and feeded to that instance, and the instance enters the "ready" state.

Protocol Execution. A protocol execution algorithm is a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the parties behave in response to signals (messages) from their environment. Upon receiving an incoming signal (message) M_{in}, an instance runs the protocol P and generates

$$(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}_U^i, \mathsf{sid}_U^i, \mathsf{pid}_U^i, \mathrm{ssk}, St_U^i) \leftarrow \mathrm{P}(1^k, U, \mathrm{pk}_U, \mathrm{sk}_U, St_U^i, M_{\mathrm{in}}).$$

The first component M_{out} corresponds to the responding message, the second component acc denotes the *decision* the instance has made, and the third component term_U^i indicates if the protocol execution has been terminated. A session id (sid_U^i), and partner id (pid_U^i) may be generated during the protocol execution. When the decision is *accept*, the instance holds a session key (ssk) which is to be used by upper layer applications. For all the protocols we analyze in this paper, we assume the state information St_U^i is erased from the memory of U once term_U^i becomes true.

Partnership. The partnership between two instances is defined via parter ID (pid) and session ID (sid). The pid names the party with which the instance believes it has just exchanged a key, and the sid is an identifier which uniquely labels the AKE session. We say two instances Π_U^i and Π_V^j are partners if $\mathsf{pid}_U^i = V, \mathsf{pid}_V^j = U$ and $\mathsf{sid}_U^i = \mathsf{sid}_V^j$.

3.2 Security Models

We define two security models to capture the two scenarios (namely, Reset-1 and Reset-2) where the randomness of an AKE protocol goes bad. However, we assume that the long-lived keys of all the honest party in the set $\mathcal{U}$ are securely generated using fresh random coins.

procedure Initialize
For all $U \in \mathcal{U}$
  $(\mathrm{pk}_U, \mathrm{sk}_U) \leftarrow\!\!\$\ \mathsf{SKG}(1^k, U)$; $T_U \leftarrow \emptyset$
Timer $\leftarrow 0$; $b \leftarrow\!\!\$\ \{0,1\}$; $\mathcal{MU} \leftarrow \emptyset$
return $\{\mathrm{pk}_U\}_{U \in \mathcal{U}}$

procedure Register(U, pk)
If $(U \in (\mathcal{U} \cup \mathcal{MU}) \vee \mathrm{pk} \in \{\mathrm{pk}_V\}_{V \in \mathcal{U} \cup \mathcal{MU}})$ then
  return *Invalid*
$\mathcal{MU} \leftarrow \mathcal{MU} \cup \{U\}$
return true

procedure NewInstance(U, i, N)
If $(U \notin \mathcal{U} \vee i \in T_U)$ then return *Invalid*
$T_U \leftarrow T_U \cup \{i\}$; $\mathrm{N}^i_U \leftarrow \mathrm{N}$; $St^i_U \leftarrow (\mathrm{N}^i_U, \mathsf{ready})$
$\mathsf{acc}^i_U \leftarrow \mathsf{false}$; $\mathsf{term}^i_U \leftarrow \mathsf{false}$
$\mathsf{sid}^i_U \leftarrow \perp$; $\mathsf{pid}^i_U \leftarrow \perp$; $\mathrm{ssk}^i_U \leftarrow \perp$
return true

procedure Send(U, i, M_{in})
If $(U \notin \mathcal{U} \vee i \notin T_U \vee \mathsf{term}^i_U)$ then return *Invalid*
$(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}^i_U, \mathsf{sid}^i_U, \mathsf{pid}^i_U, \mathrm{ssk}, St^i_U)$
  $\leftarrow \mathrm{P}(1^k, U, \mathrm{pk}_U, \mathrm{sk}_U, St^i_U, M_{\mathrm{in}})$
If $(\mathsf{acc} \wedge \text{ not } \mathsf{acc}^i_U)$ then
  $\mathrm{ssk}^i_U \leftarrow \mathrm{ssk}$; $\mathsf{acc}^i_U \leftarrow \mathsf{true}$
return $(M_{\mathrm{out}}, \mathsf{acc}, \mathsf{term}^i_U, \mathsf{sid}^i_U, \mathsf{pid}^i_U)$

procedure Reveal(U, i)
If $(U \notin \mathcal{U} \vee i \notin T_U)$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$; $\mathsf{Time}[\mathbf{Reveal}, (U, i)] \leftarrow$ Timer
return ssk^i_U

procedure Corrupt(U)
If $U \notin \mathcal{U}$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$
$\mathsf{Time}[\mathbf{Corrupt}, U] \leftarrow$ Timer
return sk_U

procedure Test(U^*, i^*)
If $U^* \notin \mathcal{U}$ then return *Invalid*
If (not $\mathsf{acc}^{i^*}_{U^*}$) then return *Invalid*
$K \leftarrow\!\!\$$ KeySpace
If $b = 0$ then return K
Else return $\mathrm{ssk}^{i^*}_{U^*}$

procedure Finalize(b')
$V^* \leftarrow \mathsf{pid}^{i^*}_{U^*}$
If $V^* \notin \mathcal{U}$ then return false
If $(\mathsf{Time}[\mathbf{Corrupt}, U^*]$
  $\vee \mathsf{Time}[\mathbf{Corrupt}, V^*])$
  return false
If $(\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)])$
  return false
If $(\exists i, i \neq i^* \wedge \mathrm{N}^i_{U^*} = \mathrm{N}^{i^*}_{U^*})$
  return false
If $(\exists j^* \in T_{V^*}, \mathsf{pid}^{j^*}_{V^*} = U^*$
  $\wedge \mathsf{sid}^{j^*}_{V^*} = \mathsf{sid}^{i^*}_{U^*})$
  If $(\exists j, j \neq j^* \wedge \mathrm{N}^j_{V^*} = \mathrm{N}^{j^*}_{V^*})$
    return false
  If $(\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)])$
    return false
return $(b = b')$

Fig. 1. Game RAKE-1.

Reset-1 Model. In this model, we consider the scenario where the randomness of each instance is completely controlled by the adversary. The formal definition is given in Figure 1 where in total six types of oracle queries are defined to capture the adversarial capabilities. In the following we explain those oracle queries in detail.

Register(U, pk_U) This oracle query allows the adversary A to register a new user U with public key pk_U. Here we only require that neither the

user identity U nor the public key pk_U exists in the system. In particular, we do not require the adversary to provide a proof of knowledge on the secret key with regard to pk_U.

NewInstance(U, i, N) This oracle query allows A to initialize a new instance Π_U^i within party U with a binary string N which serves as the random tape of Π_U^i.

Send(U, i, M_{in}) This oracle query invokes instance i of U with message M_{in}. The instance then runs $\mathrm{P}(1^k, U, \mathrm{pk}_U, \mathrm{sk}_U, St_U^i, M_{\mathrm{in}})$ and sends the response back to the adversary. Should Π_U^i terminate or accept will be made available to A. The session id sid_U^i and partner id pid_U^i are also made available to A once they are available.

Reveal(U, i) If oracle Π_U^i has accepted and generated a session key ssk_U^i, then ssk_U^i is returned to the adversary.

Corrupt(U) By making this oracle query, adversary A obtains the long-lived secret key sk_U of party U.

Test(U^*, i^*) By making this oracle query, A selects a challenge instance $\Pi_{U^*}^{i^*}$. If $\Pi_{U^*}^{i^*}$ has accepted, holding a session key $\mathrm{ssk}_{U^*}^{i^*}$, then the following happens. If the coin b, flipped in the **Initialize** phase, is 1, then $\mathrm{ssk}_{U^*}^{i^*}$ is returned to the adversary. If $b = 0$, then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game.

The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space. However, some oracle queries will render session keys exposed. By issuing these queries the adversary can trivially win the game. To exclude these trivial attacks, we consider the adversary to be successful only if it specifies a *fresh* oracle in the **Test** query.

First of all, the adversary can trivially derive a session key if one of the parties involved in that session is the adversary itself (i.e. one party is created by the adversary via a **Register** query).

The adversary will learn a party's long lived key by making a **Corrupt** query. Since in the Reset-1 model, randomness is completely controlled by the adversary, once a party is corrupted, the adversary is able to derive all state information and session keys ever generated by the party. So there is no security guarantee on session keys of any corrupted party. In other words, we don't consider the notion of *forward secrecy* in the Reset-1 model.

The adversary can certainly learn the value of a session key via a **Reveal** query. In the reset setting, the adversary can also derive a session key by mounting the *reset-and-reply* attack. Specifically, the adversary

first activates a protocol execution between instance Π_U^i with random tape N_U, and instance Π_V^j with random tape N_V. Then it activates another instance $\Pi_U^{i'}$ with the same random tape N_U. By replaying messages from Π_V^j, the adversary makes $\mathrm{ssk}_U^{i'} = \mathrm{ssk}_U^i$. In this case, revealing ssk_U^i (or using it in a upper layer application) will automatically render $\mathrm{ssk}_U^{i'}$ insecure, and vice versa. This type of attacks imply that as long as the random tape of one instance Π_U^i is used by another instance $\Pi_U^{i'}$, there is no security guarantee on the session keys generated by these two instances. So when defining the freshness of an instance, we require that its random tape is never used by another instance. Our goal is to design AKE protocols such that reset attacks would not affect the security of session keys generated by those un-reset instances.

Definition 1. *Let $\mathcal{AKE}$ be an AKE protocol. Let A be a Reset-1 adversary against $\mathcal{AKE}$ and k a security parameter. The advantage of A is defined as*

$$\mathbf{Adv}^{\text{rake-1}}_{\mathcal{AKE},A}(k) = \Pr\left[\,\text{RAKE-1}_{\mathcal{AKE},A}(k) \Rightarrow \mathsf{true}\,\right] - 1/2\,.$$

We say $\mathcal{AKE}$ is secure in the Reset-1 model if

1. *in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and*
2. *for any PPT adversary A, $\mathbf{Adv}^{\text{rake-1}}_{\mathcal{AKE},A}(k)$ is negligible.*

Reset-2 Model. In this model, we consider the scenario where the adversary is able to perform reset attacks, but unable to directly set the value of the random coins. The game RAKE-2, described in Figure 2, is used to define the security of AKE protocols in the Reset-2 setting. The definitions of oracle queries **Register**, **Send**, **Reveal**, **Corrupt** and **Test** are the same as those in game RAKE-1. But differently, when initializing a new user instance Π_U^i via a **NewInstance** query, the adversary does not set the random coins directly. Instead, it can specify another instance Π_U^j that has already been initialized, and instance Π_U^i would use the same random coins that Π_U^j has used. The adversary can also let Π_U^i use fresh random coins by setting $j = \perp$.

This adversarial model enables us to define *forward secrecy*. Recall that forward secrecy requires that compromising two users' long-lived secret keys should not allow the adversary to compromise any already established session key. We say an instance Π_U^i ($U \in \mathcal{U}$) is fs-unfresh in the Reset-2 model if any of the following conditions is true:

procedure Initialize
same as in Game RAKE-1

procedure Register(U, pk)
same as in Game RAKE-1

procedure NewInstance(U, i, j)
If $(U \notin \mathcal{U} \vee i \in T_U)$ then return *Invalid*
If $j \neq \perp \wedge j \notin T_U$ then return *Invalid*
If $j = \perp$ then $\mathrm{R}_U^i \leftarrow\!\!\$$ RandomCoins
Else $\mathrm{R}_U^i \leftarrow \mathrm{R}_U^j$
$T_U \leftarrow T_U \cup \{i\}$; $St_U^i \leftarrow (\mathrm{R}_U^i, \mathsf{ready})$
$\mathsf{acc}_U^i \leftarrow \mathsf{false}$; $\mathsf{term}_U^i \leftarrow \mathsf{false}$
$\mathsf{sid}_U^i \leftarrow \perp$; $\mathsf{pid}_U^i \leftarrow \perp$; $\mathrm{ssk}_U^i \leftarrow \perp$
return true

procedure Send(U, i, M_{in})
same as in Game RAKE-1

procedure Reveal(U, i)
same as in Game RAKE-1

procedure Corrupt(U)
same as in Game RAKE-1

procedure Test(U^*, i^*)
same as in Game RAKE-1

procedure Finalize(b')
$V^* \leftarrow \mathsf{pid}_{U^*}^{i^*}$
If $V^* \notin \mathcal{U}$ then return false
If $(\exists i, i \neq i^* \wedge \mathrm{R}_{U^*}^i = \mathrm{R}_{U^*}^{i^*})$
  return false
If $(\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)])$
  return false
If $(\exists j^* \in T_{V^*}, \mathsf{pid}_{V^*}^{j^*} = U^*$
  $\wedge \mathsf{sid}_{V^*}^{j^*} = \mathsf{sid}_{U^*}^{i^*})$
  If $(\exists j, j \neq j^* \wedge \mathrm{R}_{V^*}^j = \mathrm{R}_{V^*}^{j^*})$
    return false
  If $(\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)])$
    return false
Else
  If $(\mathsf{Time}[\mathbf{Corrupt}, V^*])$
    return false
return $(b = b')$

Fig. 2. Game RAKE-2.

1. pid_U^i is created by the adversary via a **Register** query.
2. A reveals the session key of Π_U^i.
3. There exists another instance of U whose random tape is the same as that of Π_U^i (i.e. a reset attack against Π_U^i has occured).
4. Condition 2 is true regarding the partner-oracle of Π_U^i (if it exists).
5. Condition 3 is true regarding the partner-oracle of Π_U^i (if it exists).
6. Π_U^i has no partner instance, and A corrupts pid_U^i.

Otherwise, we say Π_U^i is fs-fresh.

Definition 2. *Let $\mathcal{AKE}$ be an AKE protocol. Let A be a Reset-2 adversary against $\mathcal{AKE}$ and k a security parameter. The advantage of A is defined as*

$$\mathbf{Adv}^{\text{rake-2}}_{\mathcal{AKE},A}(k) = \Pr\left[\,\text{RAKE-2}_{\mathcal{AKE},A}(k) \Rightarrow \mathsf{true}\,\right] - 1/2\ .$$

We say $\mathcal{AKE}$ is secure in the Reset-2 model if

1. *in the presence of a benign adversary who faithfully conveys messages, then two partnering instances output the same session key; and*
2. *for any PPT adversary A,* $\mathbf{Adv}^{\text{rake-2}}_{\mathcal{AKE},A}(k)$ *is negligible.*

Strong Corruption. So far we only consider the so called "weak corruption model". To define strong corruption in our models, we follow the approach of Canetti and Krawczyk [13] and introduce a new query called **RevealState** query.

procedure RevealState(U, i)
If $(U \notin \mathcal{U} \vee i \notin T_U)$ then return *Invalid*
Timer $\leftarrow$ Timer $+ 1$; $\mathsf{Time}[\mathbf{RevealState}, (U, i)] \leftarrow$ Timer
return St^i_U

Now an additional restriction to the adversary A is that A cannot ask the **RevealState** query to the instance $\Pi^{i^*}_{U^*}$ or its partner $\Pi^{j^*}_{V^*}$ (if the latter exists).

procedure Finalize(b')
...
If $(\mathsf{Time}[\mathbf{Reveal}, (U^*, i^*)] \underline{\vee \mathsf{Time}[\mathbf{RevealState}, (U^*, i^*)]})$
return false
...
If $(\mathsf{Time}[\mathbf{Reveal}, (V^*, j^*)] \underline{\vee \mathsf{Time}[\mathbf{RevealState}, (V^*, j^*)]})$
return false
...

It is also worth noting that by adding the **RevealState** query, our Reset-2 model can be considered as Resettable CK model.

4 Resettable Security of Existing AKE Protocols

It is obvious to see that many widely deployed AKE protocols such as ISO [24], SIGMA [27, 14], JFK [2] and SKEME [28] are insecure in the Reset-1 model since for these protocols, the secrecy of the session key solely relies on the secrecy of the ephemeral secrets. Some of these protocols are insecure in the Reset-2 model either. As we have briefly mentioned before, for SIG-DH protocols, if they are implemented using DSS (or any signature scheme under the Fiat-Shamir paradigm [20]), then they are insecure in either of our reset models as the adversary can retrieve the long-lived signing key of an honest user by letting the user sign two different messages using the same randomness.

The HMQV protocol, proposed by Krawczyk in [26], is currently one of the most prominent AKE protocols. Besides achieving proven security and high efficiency, the HMQV protocol has several extra features, such as resilience to leakage of the DH exponents. However, according to a recent result by Menezes and Ustaoglu [31], an adversary can derive the long-lived secret key of an honest user if the adversary can make the user use the same randomness in different sessions, which indicates HMQV is insecure in either of our reset models.

5 From Reset-2 Security to Reset-1 and Reset-2 Security

In this section, we show that though the Reset-1 and Reset-2 models are incomparable, we can do a simple transformation on a Reset-2 secure AKE protocol to derive a new protocol that is secure in both Reset-1 and Reset-2 models.

The Transformation. Given a protocol $\Pi = (\mathsf{SKG}, \mathrm{P})$ that is secure in the Reset-2 model, and a pseudo-random function family $\mathbb{F} = \{\mathsf{F}_K : \{0,1\}^{\rho(k)} \rightarrow \{0,1\}^{\ell(k)} | K \in \{0,1\}^{\delta(k)}\}$ where $\rho(k), \ell(k)$ and $\delta(k)$ are all polynomials of k, and $\ell(k)$ denotes the maximum number of random bits needed by a party in an execution of P, we construct a new protocol $\Pi' = (\mathsf{SKG}', \mathrm{P}')$ as follows:

- $\mathsf{SKG}'(1^k)$: run $\mathsf{SKG}(1^k)$ to generate $(\mathrm{pk}, \mathrm{sk})$, select $K \leftarrow\!\!\$\ \{0,1\}^{\delta(k)}$. Set $\mathrm{pk}' = \mathrm{pk}$ and $\mathrm{sk}' = (\mathrm{sk}, K)$.
- P': get a $\rho(k)$-bit random string r, then compute $r' \leftarrow \mathsf{F}_K(r)$ and run P with random coins r'.

Theorem 1. *If Π is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and $\mathbb{F}$ is a secure pseudo-random function family, then Π' is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-1 model.*

The detailed proof is deferred to the full paper.

The pseudo-random function (PRF) family $\mathbb{F}$ is the central tool for our transformation. However, the security of a pseudo-random function $\mathsf{F}_K(\cdot)$ relies on the secrecy of the key K. When the key is known to the adversary, then we cannot assume the output of the function is still computationally indistinguishable from truly random strings. So a problem arises regarding our transformation: the resulting protocol "seems" no longer secure in the Reset-2 model. Recall in the Reset-2 model, the adversary is allowed to corrupt the long-lived key $\mathrm{sk}'_{U^*} = (\mathrm{sk}_{U^*}, K_{U^*})$ of the user U^* that output

by the adversary in the Test query, then even given a truly random string r, we cannot guarantee $\mathsf{F}_{K_{U^*}}(r)$ is random from the viewpoint of the adversary who knows K_{U^*}.

Fortunately, this problem can be resolved, but we need an extra requirement on $\mathbb{F}$, that is, we require $\mathbb{F}$ to be a Strong Randomness Extractor (SRE) [17]. In [15], Chevassut et al. showed that those very strong (i.e. the adversary has very small winning advantage) pseudo-random function families are also good strong randomness extractors. For real implementation, the HMAC function [5], which is widely used in the real practice (e.g., TLS and IKE), is a good candidate for our purpose [3, 18, 33].

Theorem 2. *If Π is a secure AKE protocol in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model, and $\mathbb{F}$ is a pseudo-random function family and a strong randomness extractor, then Π' is secure in the Weak-Corruption (Strong-Corruption, resp.) Reset-2 model.*

The proof is deferred to the full paper.

6 A New SIG-DH Protocol

In this section, we modify the ISO protocol [24, 13] to obtain a new SIG-DH protocol that is secure in both Reset-1 and Reset-2 models.

We first construct a variant of the ISO protocol, denoted by ISO-R2 (Fig. 3), that is secure in the Reset-2 model. The protocol uses a digital signature scheme $\mathcal{DS} = (\mathcal{DS}.\mathsf{SKG}, \mathcal{DS}.\mathsf{Sign}, \mathcal{DS}.\mathsf{Vf})$ that is deterministic (i.e., the signing algorithm $\mathcal{DS}.\mathsf{Sign}$ is deterministic) and existentially unforgeable under adaptive chosen-message attack (uf-cma) [21].

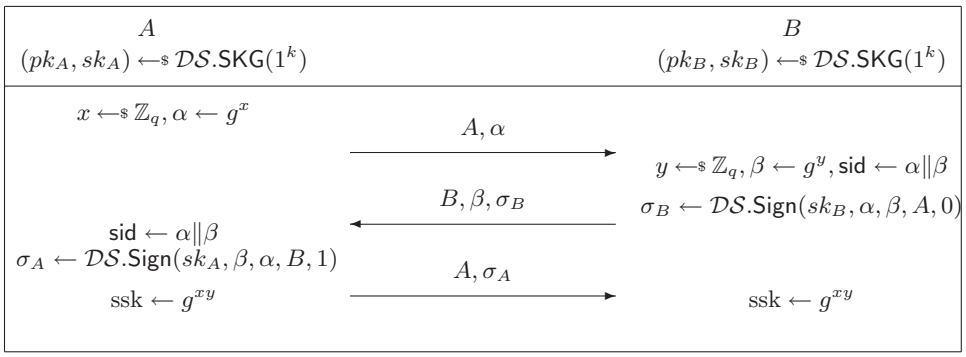

Fig. 3. The ISO-R2 Protocol.

Theorem 3. *The ISO-R2 protocol is secure in the Strong Corruption Reset-2 model if $\mathcal{DS}$ is a* uf-cma *secure deterministic digital signature scheme, and the DDH assumption holds in the underlying group.*

The proof is deferred to the full paper.

Given the ISO-R2 protocol, we can then apply the transformation in Sec. 5 to obtain a new protocol that is secure in both Reset-1 and Reset-2 models. We omit the transformed protocol here.

Acknowledgements

We thank Mihir Bellare for his suggestions on the modeling part of this paper. We also thank the reviewers for their comments and suggestions.

References

1. Digital singature standard. National Institute of Standards and Technology, NIST FIPS PUB 186, May 1994.
2. William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just fast keying: Key agreement in a hostile Internet. *ACM Trans. Inf. Syst. Secur.*, 7(2):242–273, 2004.
3. Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO 2006*, pages 602–619, 2006.
4. Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009*, pages 232–249.
5. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96*, pages 1–15, 1996.
6. Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. Identification protocols secure against reset attacks. In *EUROCRYPT 2001*, pages 495–511.
7. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO'93*, pages 232–249.
8. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution — the three party case. In *28th ACM STOC*, pages 57–66.
9. Simon Blake-Wilson and Alfred Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In *Security Protocols Workshop*, pages 137–158, 1997.
10. Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. Efficient one-round key exchange in the standard model. In *ACISP 2008*, pages 69–83. Full version available at http://eprint.iacr.org/2008/007.
11. Burton S. Kaliski Jr. An unknown key-share attack on the MQV key agreement protocol. *ACM Trans. Inf. Syst. Secur.*, 4(3):275–288, 2001.
12. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *32nd ACM STOC*, pages 235–244.
13. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, pages 453–474. `http://eprint.iacr.org/2001/040/`.

14. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *CRYPTO 2002*, pages 143–161. `http://eprint.iacr.org/2002/120/`.
15. Olivier Chevassut, Pierre-Alain Fouque, Pierrick Gaudry, and David Pointcheval. Key derivation and randomness extraction. Cryptology ePrint Archive, Report 2005/061, 2005. `http://eprint.iacr.org/`.
16. Anand Desai, Alejandro Hevia, and Yiqun Lisa Yin. A practice-oriented treatment of pseudorandom number generators. In *EUROCRYPT 2002*, pages 368–383.
17. Y. Dodis. Exposure-resilient cryptography. PhD Thesis, MIT, 2000.
18. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In *CRYPTO 2004*, pages 494–510.
19. Donald Eastlake, Steve Crocker, and Jeff Schiller. *IETF RFC 1750: Randomness Recommendations for Security*, 1994.
20. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, pages 186–194.
21. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
22. Vipul Goyal and Amit Sahai. Resettably secure computation. In *EUROCRYPT 2009*, pages 54–71.
23. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, 1998.
24. Entity authentication mechanisms - Part 3: Entity authentication using asymmetric techniques. ISO/IEC IS 9798-3, 1993.
25. C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, 2005.
26. Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO 2005*, pages 546–566.
27. Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In *CRYPTO 2003*, pages 400–425.
28. Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for Internet. In *NDSS*, pages 114–127, 1996.
29. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16, 2007.
30. Tim Matthews. Suggestions for random number generation in software. *RSA Laboratories Bulletin # 1*, January 1996.
31. Alfred Menezes and Berkant Ustaoglu. On reusing ephemeral keys in Diffie-Hellman key agreement protocols. *International Journal of Applied Cryptography*, to appear. `http://www.math.uwaterloo.ca/~ajmeneze/research.html`.
32. Tatsuaki Okamoto. Authenticated key exchange and key encapsulation in the standard model. In *Advances in Cryptology - ASIACRYPT 2007*, pages 474–484. Full paper available at http://eprint.iacr.org/2007/473.
33. Pierre-Alain Fouque and David Pointcheval and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. In *ASIACCS*, pages 21–32, 2008.
34. Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *Network and Distributed System Security Symposium (NDSS)*, 2010.
35. Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In *Topics in Cryptology - CT-RSA*, pages 41–56, 2010.