Related Randomness Attacks for Public Key Encryption*

Kenneth G. Paterson, Jacob C. N. Schuldt, Dale L. Sibborn

Information Security Group, Royal Holloway, University of London
{kenny.paterson,jacob.schuldt,dale.sibborn.2011}@rhul.ac.uk

Abstract. Several recent and high-profile incidents give cause to believe that randomness failures of various kinds are endemic in deployed cryptographic systems. In the face of this, it behoves cryptographic researchers to develop methods to immunise – to the extent that it is possible – cryptographic schemes against such failures. This paper considers the practically-motivated situation where an adversary is able to force a public key encryption scheme to reuse random values, and functions of those values, in encryption computations involving adversarially chosen public keys and messages. It presents a security model appropriate to this situation, along with variants of this model. It also provides necessary conditions on the set of functions used in order to attain this security notation, and demonstrates that these conditions are also sufficient in the Random Oracle Model. Further standard model constructions achieving weaker security notions are also given, with these constructions having interesting connections to other primitives including: pseudo-random functions that are secure in the related key attack setting; Correlated Input Secure hash functions; and public key encryption schemes that are secure in the auxiliary input setting (this being a special type of leakage resilience).

1 Introduction

Modern cryptographic primitives are heavy consumers of randomness. Unfortunately, random number generators (RNGs) used to provide this randomness often fail in practice [17, 19, 21, 22, 13, 1, 16, 27]. This is due to issues including poor algorithmic design, software bugs, insufficient or poor estimation of system entropy, and the handling of randomness across virtual machine resets [29]. The results of randomness failures can be catastrophic and newsworthy in practice – DSA, ECDSA and Schnorr private signing keys can be exposed [9, 29]; plaintext recovery for low entropy plaintext becomes possible in the the public key encryption setting; key generation processes can be severely weakened [13, 25, 23, 10]; ephemeral Diffie-Hellman keys can become predictable leading to compromise of session keys [19]; and electronic wallet security can be compromised [11].

Evidently, randomness failures are a major problem in practice. The cryptography research community has begun to address this problem only relatively recently [30, 31, 24, 2, 35, 29]. Accepting that randomness failures are endemic and unlikely to be eliminated in totality, a basic approach is to try to *hedge* against randomness failures, that is, to design cryptographic primitives that still offer a degree of security in the face of randomness failures. Work in this direction can be summarised as follows:

- For signatures, there is a folklore de-randomisation technique which neatly sidesteps security issues arising from randomness failures: simply augment the signature scheme's private key with a key for a pseudo-random function (PRF), and derive any randomness needed during signing by applying this PRF to the message to be signed; meanwhile verification proceeds as normal. This renders trivial the problem of dealing with bad randomness for signatures.
- In the private key (symmetric) encryption setting, Rogaway [30] argued for the use of nonce-based encryption, thus reducing reliance on randomness. Rogaway and Shrimpton [31] initiated the study of misuse-resistant authenticated encryption (AE), considering the residual security of AE schemes when nonces are repeated. Katz and Kamara [24] considered the security of symmetric encryption in a chosen-randomness setting, wherein the adversary has complete control over the randomness used for encryption (except for the challenge encryption which uses fresh randomness).
- In the public key encryption (PKE) setting, Bellare *et al.* [2] considered security under *chosen distribution attack*, wherein the joint distribution of message and randomness is specified by the adversary, subject to containing a reasonable amount of min entropy. The PKE scheme designer's challenge is to find a way of "extracting" this entropy in a secure way. Bellare *et al.* gave several designs for PKE schemes achieving this notion in the Random Oracle Model (ROM) and in the standard model. This is a powerful and general approach, but does have its limitations: under extreme failure conditions, the joint message-randomness distribution may simply *fail* to contain sufficient entropy, at which point all security guarantees may be lost; moreover, for technical reasons, the model in [2] requires the target public key to be hidden from the adversary until all encryption queries have been made. This is impractical in real world applications.

* A short version of this paper appeared at PKC 2014. This is the full version. All authors were supported by EPSRC Leadership Fellowship EP/H005455/1.

– Also in the PKE setting, Yilek [35], inspired by virtual machine reset attacks in [29], considered the scenario where the adversary does not know the randomness (in contrast to the chosen-randomness setting of [24]), but can instead force the reuse of random values that are otherwise well-distributed and unknown to the adversary. This is referred to in [35] as the *Reset Attack* (RA) setting. To fully reflect the reality of randomness failures in this setting, Yilek provides the adversary with the ability to encrypt chosen messages under adversarially generated public keys using the unknown but repeated random values. This makes his model very powerful, to the extent that certain trivial attacks must be excluded by assuming the adversary is *equality-pattern respecting*. In [35], Yilek also gave a general construction in which the random coins of the encryption algorithm are used as a key to a PRF, the input to the PRF is the public key concatenated with the message to be encrypted, and the output of the PRF is then used as the 'randomness' for the encryption algorithm. This is sufficient to achieve security in his RA setting. Note that the RA security model is incomparable with the CDA model of [2].
– Ristenpart and Yilek [29] studied the use of "hedging" as a general technique for protecting against broad classes of randomness failures in already-deployed systems, and implemented and benchmarked this technique in OpenSSL. Hedging in the sense of [29] involves replacing the random value r required in some cryptographic scheme with a hash of r together with other contextual information, such as a message, algorithm or unique operation identifier, etc. Their results, while applying to a variety of different randomness failure types (see in particular [29, Figure 3]), all have their security analyses restricted to the ROM.

1.1 Motivation

Inspired by the challenge of preserving security under randomness failures, we initiate the study of security for PKE in what we call the *Related Randomness Attack* (RRA) setting. Our RRA setting builds on the RA setting from [35] and brings the theory of hedging PKE against randomness failures closer to practice. As we shall see, it also has interesting connections with related key attacks for PRFs and PKE, as developed in [5, 3, 4, 6, 34], and leakage resilient cryptography (and in particular, the techniques developed in [14] to provide security for PKE in the auxiliary input setting).

In our RRA setting, the adversary can now not only force the reuse of existing random values as in the RA setting, but can also force the use of *functions of* those random values. This power is analogous to the power granted to the adversary in the Related Key Attack (RKA) setting, wherein an adversary is able to tamper with private (or secret) keys used during cryptographic operations. The RA setting arises as the special case of our RRA setting where only the identity function is allowed. The extra adversarial power in the RRA setting allows the modelling of reset attacks in which the adversary does not have an exact reset capability, but where the randomness used after a reset is in some way related to that used on previous resets. Such behaviours were observed in the experimental work in [29]. Furthermore, our RRA setting allows modelling of situations where the randomness used in a scheme comes from a PRNG which is not regularly refreshed with new entropy, but which steps forward under some deterministic state evolution function `Next` and output function `Out`; here the appropriate functions in our RRA setting would be the compositions $\mathtt{Out}(\mathtt{Next}^i(\cdot))$.

More generally, RRA security has a strong theoretical motivation as being a stepping stone towards giving the adversary enhanced control over the inputs to cryptographic algorithms – messages (in the standard PKE setting), keys (in the RKA setting), and now randomness (in our new RRA setting). It is an interesting direction for future research to develop this theme further, by examining security in a combined RKA/RRA setting, where the adversary would be able to simultaneously tamper with *all* the inputs to a PKE scheme.

1.2 Our contributions

RRA security model In this paper, we provide a strong model and security definition for PKE in the RRA setting, which we name RRA-ATK security (where ATK = CPA or CCA) . Our model is inspired by that of Yilek for the RA setting: via access to an **Enc** oracle, we allow the adversary to get arbitrary messages encrypted under arbitrary public keys, using functions ϕ of an initial set of well-distributed but unknown random values. The public keys can even be maliciously generated, and the adversary can of course know all the corresponding private keys. The adversary is tasked with winning an indistinguishability-style game, via an **LR** oracle which gives access to encryptions of left or right messages with respect to an honestly generated target public key pk^*, but again where the adversary can force the use of functions ϕ of the initial random values. When the functions ϕ are limited to coming from some set Φ, we speak of a Φ-restricted adversary.

Because the adversary may know all but one of the private keys, it can check that its challenger is behaving correctly with respect to its encryption queries. This also rules out the possibility of achieving RRA-ATK security for any randomness recovering PKE scheme, like RSA-OAEP [7] and PKE schemes based on the Fujisaki-Okamoto transformation [18]. Moreover, the encryption queries concern public keys that are outside the control of the challenger. This increases the technical challenge of achieving security in the RRA setting. This facet of the RRA setting bears comparison with the RKA setting for PKE [4, 6, 34]. In the RKA setting, the tampering via related key functions only affects the PKE scheme's private key, and so only comes into play

when simulating *decryption* queries. By contrast, it is *encryption* queries that require special treatment in our RRA setting.

Given the power of the adversary in the RRA setting, we cannot hope to achieve security against all sets of adversarial queries. So we must restrict the adversary from achieving "trivial wins". In particular, no matter what set of functions Φ the adversary uses to modify the random values, it can win simply by making the same combination of messages and functions in its **LR** quires as in its encryption queries under target public key pk^*. These must then be ruled out by identifying forbidden combinations of queries. Thus we must assume the adversary is equality-pattern respecting, for a suitable definition that extends that of [35]. However, this alone is not enough: from the RKA setting and the results of [15], we already know that certain set of functions Φ are too powerful, in allowing trivial wins for the adversary in that setting. We should not be surprised that the same is true in our RRA setting. For example, if constant functions are allowed in the RRA setting, an adversary would trivially be able to determine which of two challenge message is encrypted in a ciphertext from the **LR** oracle. Analogously to [5], we identify collision-resistance and output-unpredictability as necessary conditions on the set of functions Φ which the adversary uses to transform random values in its attack.

ROM construction We are able to show that, in the ROM, these necessary conditions on the function set Φ are actually also sufficient. More specifically, we show how to transform any IND-ATK secure PKE scheme `PKE` into a new PKE scheme `Hash-PKE` that is RRA-ATK secure for equality-pattern respecting, Φ-restricted adversaries, simply by hashing the random input together with the public key and message during encryption. In fact, this is just an application of the hedging approach from [29], and an instance of the randomized-encrypt-with-hash (REwH) scheme from [2]. Our result then shows that this approach also provides security in our new RRA setting.

Standard model constructions Having dealt with the ROM, we then turn our attention to constructions in the standard model. Reinforcing the connections to RKA security, we are able to show that any Φ-restricted RKA-PRF can be used to build a RRA-ATK secure PKE scheme for Φ-restricted adversaries, thus transferring security from the RKA setting for PRFs to the RRA setting for PKE. But the limited range of RKA-PRFs currently available in the literature [26, 3] essentially restricts the obtained RRA-ATK secure PKE scheme to a class of functions Φ consisting of *linear* or *group-induced* functions. To achieve an RRA-ATK secure PKE scheme for richer classes of functions, we must seek alternative methods of construction.

Unfortunately, we have not been able to achieve our full RRA-ATK security notion for more interesting function classes using other constructions. So we must resort to exploring alternative versions of this notion in order to make progress. We relax RRA-ATK security along two independent dimensions: the degree of control that the adversary enjoys over the public keys under which it can force encryptions for related random values, and the degree of adaptivity it has in the selection of functions $\phi \in \Phi$:

- We first consider the situation where the public keys are all honestly generated at the start of the security game, and the public keys and all but one of the private keys are then given to the adversary — the honest-key, related randomness attack (HK-RRA) setting. This is a reasonable relaxation in that, in practice, all the public keys that the adversary might be able to induce a user to encrypt under would be properly generated by users and then certified by a CA ahead of time. In this setting, we provide a generic construction for a scheme achieving HK-RRA-ATK security based on combining any IND-ATK secure PKE scheme with a Correlated-Input Secure (CIS) hash function [20]. Currently known instantiations of CIS hash functions allow us to obtain selective, HK-RRA-ATK security for Φ-restricted adversaries where Φ is a large class of polynomial functions (as opposed to the linear functions we can achieve using our RKA-PRF-based construction). Here, selectivity refers to the adversary committing at the start of the game to the set of functions it will use.
- We then consider the situation where there is no restriction on public keys, but the adversary is committed up-front to a vector of functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$ that it will use in its attack, and where security is in the end quantified over all choices of $\boldsymbol{\phi}$ from some set $\boldsymbol{\Phi}$. This quantification is subtly different from allowing the adversary a fully adaptive choice of functions $\phi \in \Phi$ (for a detailed discussion, see Section 3). In this situation, we refer to the function-vector, related randomness attack (FV-RRA) model. Here, we are able to give a direct construction for a PKE scheme that is FV-RRA-ATK secure solely under the DDH assumption, assuming the component functions ϕ_i of $\boldsymbol{\phi}$ are simultaneously hard to invert on a random input. Our scheme is inspired by a PKE scheme of Boneh *et al.* [12] that is secure in the so-called *auxiliary input setting*, wherein the adversary is given a hard-to-invert function of the secret key as part of its input. By swapping the roles of secret key and randomness in the Boneh *et al.* scheme, we are able to obtain security in a setting where a hard-to-invert function of the encryption randomness is leaked to the adversary. This leakage is then sufficient to allow us to simulate the encryptions for adversarially chosen public keys. For technical reasons, to obtain a construction, we must also limit our adversary to using the identity function when accessing its **LR** oracle.

proc. Initialise(λ):	**proc. LR**(m_0, m_1):	**proc. Dec**(c):
$b \leftarrow_\$ \{0,1\}$;	$c \leftarrow_\$ \mathtt{PKE.E}(pk, m_b)$	If $c \in \mathcal{S}$, return $\perp$
$(pk, sk) \leftarrow_\$ \mathtt{PKE.K}(1^\lambda)$;	$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$	Else return $\mathtt{PKE.D}(sk, c)$
$\mathcal{S} \leftarrow \emptyset$; Return pk	return c	
		proc. Finalise(b'):
		If $b = b'$, return 1

Fig. 1. Game IND-CCA for PKE.

To summarise, in the standard model, we can achieve our full security notion, RRA-ATK security, but only for a limited class of functions Φ (inherited from known results on RKA-PRFs), while we can achieve alternative security notions for richer classes Φ.

1.3 Future Directions

In this paper, we concentrate on PKE, but RRA security notions can be developed for other primitives. As previously noted, the case of signatures is quite simple, provided one is prepared to extend a scheme's private key. We would expect symmetric key encryption and key exchange primitives to be more complex. Also as noted above, our RRA setting is related to the RKA setting, and it is an open problem to develop these connections further, possibly by considering a combined RKA/RRA setting.

2 Preliminaries

Throughout the paper we will use $\lambda \in \mathbb{N}$ to denote the security parameter, which will sometimes be written in its unary representation, 1^λ. We denote by $y \leftarrow x$ the assignment of y to x, and by $s \leftarrow_\$ S$ we denote the selection of an element s uniformly at random from the set S. The notation $[n]$ represents the set $\{1, 2, \ldots, n\}$. We let $\mathrm{FF}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ denote the set of all families of functions $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$.

All our security games and proofs will utilise code-based games and the associated language. Here we briefly recall the basic definitions from [8]. A game consists of at least two procedures. We begin with **Initialise**, which assigns starting values to all variables and then gives outputs, if there are any, to the adversary. The adversary $\mathcal{A}$ may then submit queries to the oracle procedures, and when $\mathcal{A}$ halts (and possibly outputs a value) the **Finalise** procedure begins. **Finalise** will take the output from $\mathcal{A}$ (if there is one) as its input and will output its own value. The value output by **Finalise** is defined to be the output of the game. We write $\mathbb{P}[G^{\mathcal{A}} \Rightarrow b]$ to denote the probability that game G outputs bit b when run with $\mathcal{A}$. However, in some proofs we will use the notation $\mathcal{A}^G \Rightarrow b$. This means that the adversary outputs b when run with the game G. We will occasionally use the notation $\mathcal{A}(x) \Rightarrow b$, which denotes adversary $\mathcal{A}$ outputting b when given the input x. For brevity, in what follows ATK will denote either CPA or CCA, where theorems or statements apply to both games. Any proofs or figures will refer to the CCA setting, but may be easily modified to the CPA case.

2.1 Public Key Encryption

We denote a specific PKE scheme by $\mathtt{PKE} = (\mathtt{PKE.K}, \mathtt{PKE.E}, \mathtt{PKE.D})$. All three algorithms are polynomial-time. The randomised key generation algorithm $\mathtt{PKE.K}$ takes the security parameter as its input and outputs a key pair (pk, sk). The encryption algorithm, on input a message $m \in \mathcal{M}$ and a public key pk chooses random coins from $\mathtt{Rnd}$ and uses these coins to output a ciphertext c. The decryption algorithm is deterministic. Its inputs are a private key sk and a ciphertext c. The algorithm either outputs a message m or an error symbol $\perp$. We require the scheme $\mathtt{PKE}$ to satisfy the correctness property. That is, for all $\lambda \in \mathbb{N}$, all all pairs (pk, sk) output by the key generation algorithm, and all messages $m \in \mathcal{M}$, we require that $\mathtt{PKE.D}(sk, \mathtt{PKE.E}(pk, m)) = m$.

Definition 1. *The advantage of an* IND-ATK *adversary $\mathcal{A}$ against a scheme* $\mathtt{PKE}$ *is*

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathtt{PKE},\mathcal{A}}(\lambda) := 2 \cdot \mathbb{P}[\text{IND-ATK}^{\mathcal{A}}_{\mathtt{PKE}}(\lambda) \Rightarrow 1] - 1$$

where game IND-ATK *is shown in Figure 1. A scheme* $\mathtt{PKE}$ *is* IND-ATK *secure if the advantage of any polynomial-time adversary is negligible in the security parameter λ.*

2.2 Data Encapsulation Mechanisms

We define a DEM $\mathtt{DEM} = (\mathtt{DEM.K}, \mathtt{DEM.E}, \mathtt{DEM.D})$ in the natural way. The three algorithms are polynomial-time. Key generation takes the security parameter as an input and outputs a key K. The encryption algorithm takes a message $m \in \mathcal{M}^{\mathtt{DEM}}_\lambda$ and a key K as its inputs, chooses random coins from $\mathtt{Rnd}^{\mathtt{DEM}}_\lambda$, and outputs a ciphertext c. Decryption takes a key K and a ciphertext c. The output is either a message m or an error symbol $\perp$. Again we require the correctness property, so that for all K output by the key generation algorithm and all m, we have $\mathtt{DEM.D}(K, \mathtt{DEM.E}(K, m)) = m$.

proc. Initialise(λ):
$b \leftarrow_{\$} \{0,1\}$;
$K \leftarrow_{\$} \mathtt{DEM.K}(1^{\lambda})$;
$\mathcal{S} \leftarrow \emptyset$

proc. LR(m_0, m_1):
$c \leftarrow_{\$} \mathtt{DEM.E}(K, m_b)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$
return c

proc. Dec(c):
If $c \in \mathcal{S}$, return $\perp$
Else return $\mathtt{DEM.D}(K, c)$

proc. Finalise(b'):
If $b = b'$, return 1

Fig. 2. Game IND-CCA for a DEM DEM.

proc. Initialise(λ):
$K \leftarrow_{\$} \mathtt{Keys}_{\lambda}$

proc. Function(x):
Return $F(K, x)$.

proc. Finalise(b):
return b

proc. Initialise(λ):
FunTab $\leftarrow \emptyset$

proc. Function(x):
If FunTab[x] $=\perp$, then
  FunTab[x] $\leftarrow_{\$} \mathtt{Rng}_{\lambda}$
Return FunTab[x].

proc. Finalise(b):
return b

Fig. 3. Games for PRF security. Game PRFReal is on the left, PRFRand on the right.

Definition 2. *The advantage of an* IND-ATK *adversary* $\mathcal{A}$ *against a scheme* DEM *is*

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathtt{DEM},\mathcal{A}}(\lambda) := 2 \cdot \mathbb{P}[\text{IND-ATK}^{\mathcal{A}}_{\mathtt{DEM}}(\lambda) \Rightarrow 1] - 1$$

where game IND-ATK *is shown in Figure 2. A scheme* DEM *is* IND-ATK *secure if the advantage of any polynomial-time adversary is negligible in the security parameter* λ.

2.3 Pseudorandom Functions

Definition 3. *Let* $F : \mathtt{Keys}_{\lambda} \times \mathtt{Dom}_{\lambda} \rightarrow \mathtt{Rng}_{\lambda}$ *be a family of functions. The advantage of a* PRF *adversary* $\mathcal{A}$ *against* F *is*

$$\mathbf{Adv}^{\text{prf}}_{F,\mathcal{A}}(\lambda) := \mathbb{P}[\text{PRFReal}^{\mathcal{A}}_{F}(\lambda) \Rightarrow 1] - \mathbb{P}[\text{PRFRand}^{\mathcal{A}}_{\$}(\lambda) \Rightarrow 1]$$

where the games PRFReal *and* PRFRand *are defined in Figure 3. We say* F *is a secure* PRF *family if the advantage of any polynomial-time adversary is negligible in the security parameter* λ.

Related Key Attack Pseudorandom Function

Definition 4. *Let* $F : \mathtt{Keys}_{\lambda} \times \mathtt{Dom}_{\lambda} \rightarrow \mathtt{Rng}_{\lambda}$ *be a family of functions. The advantage of an* RKA-PRF *adversary* $\mathcal{A}$ *against* F *is*

$$\mathbf{Adv}^{\text{rka-prf}}_{F,\mathcal{A}}(\lambda) := \mathbb{P}[\text{RKA-PRFReal}^{\mathcal{A}}_{F}(\lambda) \Rightarrow 1] - \mathbb{P}[\text{RKA-PRFRand}^{\mathcal{A}}_{\$}(\lambda) \Rightarrow 1]$$

where the games PRFReal *and* PRFRand *are defined in Figure 4. We say* F *is* Φ-RKA-PRF *secure if the advantage of any* Φ*-restricted, polynomial-time adversary is negligible in the security parameter* λ.

2.4 Key Derivation Functions

Definition 5. *Let* λ *be a security parameter. Consider a polynomial-time computable function* f *that maps from* $\mathcal{D}(\lambda)$ *to* $\mathcal{R}(\lambda)$. *The advantage of a* KDF *adversary* $\mathcal{A}$ *against* f *is*

$$\mathbf{Adv}^{\text{kdf}}_{f,\mathcal{A}}(\lambda) := \mathbb{P}[\text{KDFReal}^{\mathcal{A}}_{f}(\lambda) \Rightarrow 1] - \mathbb{P}[\text{KDFRand}^{\mathcal{A}}_{\$}(\lambda) \Rightarrow 1]$$

proc. Initialise(λ):
$K \leftarrow_{\$} \mathtt{Keys}_{\lambda}$

proc. Function(ϕ, x):
return $F(\phi(K), x)$.

proc. Finalise(b):
return b

proc. Initialise(λ):
$G \leftarrow \text{FF}(\mathtt{Keys}_{\lambda}, \mathtt{Dom}_{\lambda}, \mathtt{Rng}_{\lambda})$
$K \leftarrow_{\$} \mathtt{Keys}_{\lambda}$

proc. Function(ϕ, x):
return $G(\phi(K), x)$.

proc. Finalise(b):
return b

Fig. 4. Games for RKA-PRF security. Game RKA-PRFReal is on the left, RKA-PRFRand on the right.

proc. Oracle:	**proc. Oracle**:
$s \leftarrow_\$ \mathcal{D}(\lambda)$	$r \leftarrow_\$ \mathcal{R}(\lambda)$
return $f(s)$	return r
proc. Finalise(b):	**proc. Finalise**(b):
output b	output b

Fig. 5. Games for KDF security. Game KDFReal is on the left, KDFRand is on the right.

where games KDFReal *and* KDFRand *are defined in Figure 5. We say f is a secure Key Derivation Function (KDF) if the advantage of any polynomial-time adversary is negligible in the security parameter λ.*

This definition can be extended to allow the adversary to make multiple oracle queries. In the real game, for query i, a random value $s_i \leftarrow_\$ \mathcal{D}(\lambda)$ is chosen for each query and the adversary is given $f(s_i)$. In the random game, $r_i \leftarrow_\$ \mathcal{R}(\lambda)$ is chosen for each query and the adversary is given r_i. Then, using a hybrid argument, we can show that, for any t-query adversary $\mathcal{A}$, there is a (single-query) KDF adversary $\mathcal{B}$ such that:

$$\mathbf{Adv}^{t\text{-kdf}}_{f,\mathcal{A}}(\lambda) \leq t \cdot \mathbf{Adv}^{\text{kdf}}_{f,\mathcal{B}}(\lambda)$$

where $\mathbf{Adv}^{t\text{-kdf}}_{f,\mathcal{A}}(\lambda)$ denotes the advantage of t-query KDF adversary $\mathcal{A}$ against f.

2.5 Computational Assumption

Definition 6 (Decisional q-Diffie-Hellman Inversion (q-DDHI) Problem). *The advantage of an algorithm $\mathcal{A}$ in solving the decisional q-Diffie-Hellman inversion problem in a cyclic group $\mathbb{G}$ of order $p = p(\lambda)$, is*

$$\mathbf{Adv}^{q\text{-}ddhi}_{\mathbb{G},\mathcal{A}}(\lambda) := |\mathbb{P}[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, g^{1/x}) = 1] - \mathbb{P}[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, R) = 1]|$$

where g is a random generator of $\mathbb{G}$, $x \leftarrow_\$ \mathbb{Z}_p$, $R \leftarrow_\$ \mathbb{G}$, and the probability is taken over the choice of $g, R \in \mathbb{G}$, choice of $x \in \mathbb{Z}_p$, and the random coins consumed by $\mathcal{A}$.

3 Related Randomness Security for Public Key Encryption

We now formalise our notions of related randomness security for PKE. We give a detailed treatment of our strongest notion, before sketching restricted versions. The description of our security notions will utilise code-based games and the associated language (see [8]).

Our strongest security notion, RRA-CCA security, is defined via the game in Figure 6. Here, a challenge key pair (pk^*, sk^*) for a PKE scheme $\mathtt{PKE} = (\mathtt{PKE.K}, \mathtt{PKE.E}, \mathtt{PKE.D})$ with randomness space $\mathtt{Rnd}$ is honestly generated, and the adversary is considered successful if it wins an indistinguishability game with respect to messages encrypted under pk^*. Extending the standard PKE setting, the adversary is able to control which one of polynomially many random values $r_i \in \mathtt{Rnd}$ is used in responding to each encryption query for pk^*; furthermore, the adversary is able to obtain the encryption of messages of its choice under (possibly maliciously generated) arbitrary public keys. Extending the model of Yilek [35], our adversary not only specifies which one of the random values r_i is to be used in each query, but also specifies, for each query he makes, a function ϕ on $\mathtt{Rnd}$; the value $\phi(r_i)$ is used for encryption in place of r_i. In the CCA setting, the adversary also has access to a regular decryption oracle for private key sk^*. Note that if the adversary uses *only* the identity function, then we recover the Resettability Attack (RA) model of Yilek [35].

It is not difficult to see that, as in the RA setting, an adversary may trivially win this game if no restrictions are placed on oracle queries.[1] We will shortly introduce an *equality-pattern respecting* definition for adversaries, designed to prevent trivial wins of this kind. This extends the related RA definition from [35]. However, restrictions on the functions ϕ will also be required. To illustrate the issue, consider as an extreme case the constant function ϕ_C (with $\phi_C(r) = C$ for all $r \in \mathtt{Rnd}$). Suppose the adversary submits **LR** query (m_0, m_1, j, ϕ_C) for any $m_0 \neq m_1$ and any $j \in \mathbb{N}$; the adversary receives a ciphertext c^* and then computes $c_0 = \mathtt{PKE.E}(pk^*, m_0; C)$; the adversary outputs guess $b' = 0$ if and only if $c^* = c_0$. It is easy to see that this adversary wins the RRA-ATK game with probability 1. This example is analogous to one in the related key attack setting for PRFs in [5]. Hence, we will need to restrict the class of functions which the adversary is allowed to access in its queries to come from some set Φ, in which case we speak of Φ-restricted adversaries. We have already seen that constant

[1] For example, if an adversary requests the encryption of m under the target public key using coins $\phi(r_i)$, $\mathtt{PKE.E}(pk^*, m; \phi(r_i))$, and submits **LR** query (m, m', i, ϕ), then the adversary guesses b is 0 if the two ciphertexts match, otherwise he guesses b is 1. This adversary wins the game with probability 1. As in the RA setting, such wins are unavoidable in our setting since encryption essentially becomes deterministic when the same random coins and functions ϕ are used.

proc. Initialise(λ):
$b \leftarrow_{\$} \{0,1\}$;
$(pk^*, sk^*) \leftarrow_{\$} \mathtt{PKE.K}(1^\lambda)$;
CoinTab $\leftarrow \emptyset$;
$\mathcal{S} \leftarrow \emptyset$; Return pk^*

proc. Dec(c):
If $c \in \mathcal{S}$, then return $\perp$
Else return $\mathtt{PKE.D}(sk^*, c)$

proc. LR(m_0, m_1, i, ϕ):
If CoinTab$[i] = \perp$
  CoinTab$[i] \leftarrow_{\$}$ Rnd
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \mathtt{PKE.E}(pk^*, m_b; \phi(r_i))$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$
Return c

proc. Enc(pk, m, i, ϕ):
If CoinTab$[i] = \perp$
  CoinTab$[i] \leftarrow_{\$}$ Rnd
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \mathtt{PKE.E}(pk, m; \phi(r_i))$
Return c

proc. Finalise(b'):
If $b = b'$, return 1

Fig. 6. Game RRA-ATK. (Note that if ATK = CPA, then the adversary's access to **proc. Dec** is removed.)

functions must be excluded from Φ if we are to have any hope of achieving our related randomness security notion.

Thus we have two sets of constraints that we need to consider to prevent trivial wins: those on messages and randomness indices (analogous to the RA setting from [35]) and those on functions ϕ (analogous to the RKA setting for PRFs from [5]). Let us deal with the first set of constraints first and define what it means for an adversary to be equality-pattern respecting. The following definition is adapted from [35] for our purposes.

Definition 7. *Let $\mathcal{A}$ be a Φ-restricted adversary in Game* RRA-ATK *that queries r different randomness indices to its **LR** and **Enc** oracles and makes $q_{i,\phi}$ queries to its **LR** oracle with index i and function $\phi \in \Phi$. Let $E_{i,\phi}$ be the set of all messages m such that $\mathcal{A}$ makes **Enc** query (pk^*, m, i, ϕ). Let $(m_0^{i,\phi,1}, m_1^{i,\phi,1}), \ldots, (m_0^{i,\phi,q_{i,\phi}}, m_1^{i,\phi,q_{i,\phi}})$ be $\mathcal{A}$'s **LR** queries for index $i \in [r]$ and $\phi \in \Phi$. Suppose that for all pairs $(i,\phi) \in [r] \times \Phi$ and for all $j \neq k \in [q_{i,\phi}]$, we have:*

$$m_0^{i,\phi,j} = m_0^{i,\phi,k} \text{ iff } m_1^{i,\phi,j} = m_1^{i,\phi,k}$$

and that, for all pairs $(i,\phi) \in [r] \times \Phi$, and for all $j \in [q_{i,\phi}]$, we have:

$$m_0^{i,\phi,j} \notin E_{i,\phi} \wedge m_1^{i,\phi,j} \notin E_{i,\phi}.$$

Then we say that $\mathcal{A}$ is equality-pattern respecting.

Notice that if the adversary is restricted to using only the identity function, then this definition reduces to the equality-pattern respecting definition for the RA setting, cf. [35, Appendix A].

Definition 8. *We define the advantage of an equality-pattern respecting,* RRA-ATK *adversary $\mathcal{A}$ against a PKE scheme* PKE *to be:*

$$\mathbf{Adv}^{\text{rra-atk}}_{\mathtt{PKE},\mathcal{A}}(\lambda) := 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\mathtt{PKE}}(\lambda) \Rightarrow 1] - 1.$$

A PKE scheme PKE *is said to be Φ-*RRA-ATK *secure if the advantage of any Φ-restricted, equality-pattern respecting,* RRA-ATK *adversary against* PKE *that runs in polynomial time is negligible in the security parameter λ.*

3.1 Alternative security notions

The above definition for Φ-RRA-ATK security is very powerful: it allows an adversary to submit *any* public key to its encryption oracle and allows the adversary to *adaptively* choose the functions ϕ, the only restriction being that they lie in Φ. In Theorem 3 we will exhibit conditions that are both necessary and sufficient for achieving security in this sense in the ROM (given a starting PKE scheme that satisfies the usual definition of IND-ATK security). In the standard model, we will give a construction that relies on RKA-PRFs. Since constructions for these are currently very limited in terms of the function classes they can handle, we will now consider alternative versions of the Φ-RRA-ATK notion.

The first alternative notion we consider is called *Honest Key Related Randomness* (HK-RRA) security. The security game is given in Figure 7 and has two parameters, λ and ℓ. Informally, the game itself generates a polynomial number ℓ of key pairs and returns the public keys to the adversary. The adversary then chooses which public key he wishes to be the target key, and is given the private keys corresponding to all the non-target public keys. Meanwhile, the adversary's queries to its **Enc** oracle are restricted to using the public keys generated by the game. Suitable Φ-HK-RRA-ATK security notions follow by analogy with our earlier definitions.

One may consider notions intermediate between Φ-RRA-ATK security and Φ-HK-RRA-ATK security. For example, a registered key notion could be defined, in which the adversary chooses and registers key pairs (pk, sk), with registration involving a test for validity by some procedure, and all queries involve only registered public keys. One may also consider weaker variants of these notions in which the adversary's choice of functions ϕ is non-adaptive (or *selective*). That is, the adversary must submit a set of functions $\{\phi\} \subset \Phi$ of polynomial size

proc. Initialise(λ, ℓ):
$b \leftarrow_\$ \{0,1\}$;
$\texttt{funchoice} \leftarrow \texttt{false}$;
Keys $\leftarrow \emptyset$; Functions $\leftarrow \emptyset$;
$\texttt{target} \leftarrow \texttt{false}$;
CoinTab $\leftarrow \emptyset$;
$\mathcal{S} \leftarrow \emptyset$; $(pk^*, sk^*) \leftarrow \emptyset$
For $i = 1$ to ℓ
  $(pk_i, sk_i) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$
  Keys $\leftarrow$ Keys $\cup\, pk_i$
Return Keys

proc. Target(j):
If $\texttt{target} = \texttt{true}$, return $\perp$
Else $(pk^*, sk^*) \leftarrow (pk_j, sk_j)$
$\texttt{target} \leftarrow \texttt{true}$
Return $\{sk_i\}_{i \neq j}$

proc. Enc(pk, m, i, ϕ):
If $\texttt{target} = \texttt{false}$, return $\perp$
If $pk \notin$ Keys, return $\perp$
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \texttt{PKE.E}(pk, m; \phi(r_i))$
Return c

proc. Dec(c):
If $\texttt{target} = \texttt{false}$, return $\perp$
If $c \in \mathcal{S}$, then return $\perp$
Else return $\texttt{PKE.D}(sk^*, c)$

proc. LR(m_0, m_1, i, ϕ):
If $\texttt{target} = \texttt{false}$,
  return $\perp$
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \texttt{PKE.E}(pk^*, m_b; \phi(r_i))$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$
Return c

proc. Finalise(b'):
If $b = b'$, return 1

Fig. 7. Game ℓ-HK-RRA-ATK. (Note that if ATK = CPA, then the adversary's access to **proc. Dec** is removed.)

proc. Initialise(λ):
$b \leftarrow_\$ \{0,1\}$;
$(pk^*, sk^*) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$;
CoinTab $\leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$;
return pk^*

proc. Dec(c):
If $c \in \mathcal{S}$, then return $\perp$
Else return $\texttt{PKE.D}(sk^*, c)$

proc. LR(m_0, m_1, i):
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \texttt{PKE.E}(pk^*, m_b; r_i)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$
return c

proc. Enc(pk, m, i, j):
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \texttt{PKE.E}(pk, m; \phi_j(r_i))$
return c

proc. Finalise(b'):
If $b = b'$, return 1

Fig. 8. Game $\boldsymbol{\phi}$-FV-RRA-ATK, where $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$. (As usual, if ATK = CPA, then the adversary's access to **proc. Dec** is removed.)

to the game before he is allowed to see the target public key (or set of public keys, if playing in the Honest Key setting). In this setting, we refer to Φ-sHK-RRA-ATK security.

The final alternative notion we consider is called *Function-Vector Related Randomness* (FV-RRA) security, and is based on the game in Figure 8. Here, the adversary is parameterised by a vector of functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$, and is limited to using only these functions in its oracle queries. Additionally, we restrict the adversary by demanding that the **LR** queries use only the identity function. However, once again, the adversary has complete freedom over public keys submitted to its encryption oracle. Furthermore, security will be quantified over *all* choices of vector from a particular class. (Specifically, in our construction in Section 7, we will demand that security holds over all vectors $\boldsymbol{\phi}$ that are simultaneously hard to invert on a common random input r.) This quantification actually makes our notion rather strong.

Definition 9. *Let $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$ be a vector of $q := q(\lambda)$ functions. We define the advantage of an equality-pattern respecting, $\boldsymbol{\phi}$-*FV-RRA-ATK *adversary $\mathcal{A}$ against a PKE scheme* PKE *to be:*

$$\mathbf{Adv}^{\boldsymbol{\phi}\text{-fv-rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) := 2 \cdot \mathbb{P}[\boldsymbol{\phi}\text{-FV-RRA-ATK}^{\mathcal{A}}_{\texttt{PKE}}(\lambda) \Rightarrow 1] - 1.$$

If $\boldsymbol{\Phi}$ is a set of vectors of functions, then a PKE scheme PKE *is said to be $\boldsymbol{\Phi}$-*FV-RRA-ATK *secure if, for all $\boldsymbol{\phi} \in \boldsymbol{\Phi}$, the advantage of any equality-pattern respecting, $\boldsymbol{\phi}$-*FV-RRA-ATK *adversary against* PKE *that runs in polynomial time is negligible in the security parameter λ.*

Comparison of security notions The first alternative security notion, HK-RRA-ATK security, is easily seen to be a strictly weaker notion than full RRA-ATK security[2]. Likewise, the selective models are easily seen to be weaker then their adaptive counterparts. However, the relation between full RRA-ATK security and FV-RRA-ATK security is not immediately obvious. Aside from the restriction on **LR**-queries in FV-RRA-ATK security, there is a subtle distinction between requiring security for all vectors $\boldsymbol{\phi}$ of functions from a particular set Φ and requiring security for a fully adaptive choice of functions $\phi \in \Phi$. In particular, the former notion will allow a security reduction to consider multiple runs of an adversary with different random coins for a fixed

[2] A separation can be established by considering a scheme where public keys generated by the key generation algorithm always have a certain bit set to 0, and where the encryption algorithm, given a public key with this bit set to 1 (i.e. a maliciously generated public key), will expose the randomness used for the encryption.

proc. Initialise(λ):
$(pk^*, sk^*) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$;
CoinTab $\leftarrow \emptyset$;
$\mathcal{S} \leftarrow \emptyset$; Return pk^*

proc. Enc(pk, m, i, ϕ):
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
$c \leftarrow \texttt{PKE.E}(pk, m; \phi(r_i))$
Return c

proc. LR(m_0, m_1, i, ϕ):
If CoinTab$[i] = \perp$,
  CoinTab$[i] \leftarrow_\$ \texttt{Rnd}$
$r_i \leftarrow$ CoinTab$[i]$
If $i \le q_r - j$,
  return $c \leftarrow \texttt{PKE.E}(pk^*, m_0; \phi(r_i))$
If $i > q_r - j$,
  return $c \leftarrow \texttt{PKE.E}(pk^*, m_1; \phi(r_i))$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$
return c

proc. Dec(c):
If $c \in \mathcal{S}$, then return $\perp$
Else return $\texttt{PKE.D}(sk^*, c)$.

proc. Finalise(b'):
If $b = b'$, return 1.

Fig. 9. The game G_j used in the proof of Lemma 1.

choice of function vector $\boldsymbol{\phi}$, whereas the latter notion will leave open the possibility that an adversary will chose a different sequence of functions ϕ in each run. Also note that FV-RRA-ATK security guarantees that there is no choice of $\boldsymbol{\phi}$ for which the considered scheme is weak, even if this choice might be computationally hard for an adaptive adversary to find. Furthermore, the relation between the notions might also be influenced by the considered class of functions Φ. It remains future work to fully explore and categorise the possible notions of RRA security.

It is not hard to see that our RRA security notions are incomparable with the CDA security notions of [2]. In the RA setting, Yilek defines only an equivalent of our full RRA-ATK notion; it is clear that RRA-ATK security is stronger than his RA-ATK security whenever the function set Φ contains the identity function. The same would carry over to relaxed versions of RA-ATK security.

3.2 A Simplifying Lemma

Lemma 1. *Consider an equality-pattern respecting,* RRA-ATK *adversary $\mathcal{A}$ that queries q_r distinct randomness indices and makes at most q_{LR} **LR** queries. Then there exists an equality-pattern respecting,* RRA-ATK *adversary $\mathcal{B}$ that queries at most* 1 *randomness index and makes at most q_{LR} **LR** queries such that*

$$\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) \le q_r \cdot \mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{B}}(\lambda),$$

where $\mathcal{B}$ runs in approximately the same time as $\mathcal{A}$. In the CCA setting, $\mathcal{B}$ makes the same number of decryption queries as $\mathcal{A}$.

Proof. To keep the notation from becoming too cluttered, we denote by $\mathcal{A}^{G(q_{LR},q_r)}$ an adversary $\mathcal{A}$ playing the RRA-ATK game that makes q_{LR} **LR** queries and queries q_r distinct randomness indices. To prove the lemma we will use an alternative, but equivalent, notion of adversarial advantage, namely:

$$\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) = |\mathbb{P}[\mathcal{A}^{G(q_{LR},q_r)} \Rightarrow 1 \mid b = 0] - \mathbb{P}[\mathcal{A}^{G(q_{LR},q_r)} \Rightarrow 1 \mid b = 1]|.$$

Notice that we are now interested in the output of the adversary, rather than the output of the game. Let G_0 denote the RRA-ATK security game in Figure 6 where $b = 0$. Let G_{q_r} denote the same game where $b = 1$. If games G_j are as defined in Figure 9, then

$$\begin{aligned}
\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) &= |\mathbb{P}[\mathcal{A}^{G_0} \Rightarrow 1] - \mathbb{P}[\mathcal{A}^{G_{q_r}} \Rightarrow 1]| \\
&= \left| \sum_{j=0}^{q_r-1} \mathbb{P}[\mathcal{A}^{G_j} \Rightarrow 1] - \mathbb{P}[\mathcal{A}^{G_{j+1}} \Rightarrow 1] \right| \\
&\le \sum_{j=0}^{q_r-1} |\mathbb{P}[\mathcal{A}^{G_j} \Rightarrow 1] - \mathbb{P}[\mathcal{A}^{G_{j+1}} \Rightarrow 1]|,
\end{aligned}$$

Without loss of generality, we will assume that games j^* and $j^* + 1$ have the largest difference. Then,

$$\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) \le q_r \cdot |\mathbb{P}[\mathcal{A}^{G_{j^*}} \Rightarrow 1] - \mathbb{P}[\mathcal{A}^{G_{j^*+1}} \Rightarrow 1]|.$$

The only difference between games j^* and $j^* + 1$ is in how the **LR** oracle responds to a query with randomness index $q_r - j^*$. If $\mathcal{A}$ can distinguish between games j^* and $j^* + 1$, then we can use this adversary to build an adversary $\mathcal{B}$ winning the RRA-ATK game and using only 1 randomness index. The **Initialise** procedure for $\mathcal{B}$'s RRA-ATK game is run, returning a target public key pk^* to $\mathcal{B}$. Then adversary $\mathcal{B}$ sets up the simulation for $\mathcal{A}$.

Setup
CoinTab $\leftarrow \emptyset$;
$\mathcal{S} \leftarrow \emptyset$

Then $\mathcal{B}$ forwards the key pk^* to $\mathcal{A}$. Adversary $\mathcal{B}$ will simulate either game G_{j^*} or G_{j^*+1} for $\mathcal{A}$ by answering $\mathcal{A}$'s oracle queries as follows:

Enc query (pk, m, l, ϕ)
If $l = q_r - j^*$, $\mathcal{B}$ submits (pk, m, ϕ) to its **Enc** oracle. It returns the result to $\mathcal{A}$.
Otherwise, if CoinTab$[l] = \perp$, $\mathcal{B}$ chooses CoinTab$[l] \leftarrow_{\$}$ Rnd
$r_l \leftarrow$ CoinTab$[l]$, and $\mathcal{B}$ returns $\mathtt{PKE.E}(pk, m; \phi(r_l))$.
LR query (m_0, m_1, l, ϕ)
If $l = q_r - j^*$, $\mathcal{B}$ submits (ϕ, m_0, m_1) to its **LR** oracle and returns the result to $\mathcal{A}$.
Otherwise, if CoinTab$[l] = \perp$, $\mathcal{B}$ chooses CoinTab$[l] \leftarrow_{\$}$ Rnd
$r_l \leftarrow$ CoinTab$[l]$
if $l < q_r - j^*$, $\mathcal{B}$ returns $\mathtt{PKE.E}(pk^*, m_0; \phi(r_l))$
else if $l > q_r - j^*$, $\mathcal{B}$ returns $\mathtt{PKE.E}(pk^*, m_1; \phi(r_l))$
$\mathcal{B}$ updates $\mathcal{S}$ to include the ciphertext returned to $\mathcal{A}$
Dec query c
If $c \in \mathcal{S}$, then $\mathcal{B}$ returns $\perp$.
Otherwise $\mathcal{B}$ submits c to its **Dec** oracle and the output is returned to $\mathcal{A}$.

When $\mathcal{A}$ halts with output bit b', $\mathcal{B}$ halts and outputs the same bit b'.
When $b = 0$ (and $\mathcal{B}$ receives an encryption of m_0), it perfectly simulates G_{j^*} for $\mathcal{A}$. When $b = 1$, $\mathcal{B}$ provides a perfect simulation of G_{j^*+1}. It follows that

$$\begin{aligned}\mathbf{Adv}^{\text{rra-atk}}_{\mathtt{PKE},\mathcal{A}}(\lambda) &\leq q_r \cdot |\mathbb{P}[\mathcal{A}^{G_{j^*}} \Rightarrow 1] - \mathbb{P}[\mathcal{A}^{G_{j^*+1}} \Rightarrow 1]| \\ &= q_r \cdot |\mathbb{P}[\mathcal{B} \Rightarrow 1 | b = 1] - \mathbb{P}[\mathcal{B} \Rightarrow 1 | b = 0]| \\ &= q_r \cdot \mathbf{Adv}^{\text{rra-atk}}_{\mathtt{PKE},\mathcal{B}}(\lambda).\end{aligned}$$

This completes the proof.

The above lemma shows that we need only consider adversaries that use one randomness index. The lemma actually applies for all variations of related randomness security considered above, not just our strongest RRA-ATK notion. This enables us to make a simplifying step at the beginning of all our proofs (at the cost of a q_r factor in all advantages), and to use the following simplified equality-pattern definition in all our proofs:

Definition 10. *Let $\mathcal{A}$ be a Φ-restricted adversary in Game* RRA-ATK *that queries* 1 *randomness index (assumed to be $j = 1$) to its **LR** and **Enc** oracles and makes q_ϕ queries to its **LR** oracle with function ϕ. Let E_ϕ be the set of all messages m such that $\mathcal{A}$ makes **Enc** query $(pk^*, m, 1, \phi)$. Let $(m_0^{\phi,1}, m_1^{\phi,1}), \ldots, (m_0^{\phi,q_\phi}, m_1^{\phi,q_\phi})$ be $\mathcal{A}$'s **LR** queries with function ϕ and randomness index* 1*. Suppose that for all $\phi \in \Phi$ and for all $j \neq k \in [q_\phi]$, we have:*

$$m_0^{\phi,j} = m_0^{\phi,k} \text{ iff } m_1^{\phi,j} = m_1^{\phi,k}$$

and that, for all $\phi \in \Phi$ and for all $j \in [q_\phi]$, we have:

$$m_0^{\phi,j} \notin E_\phi \wedge m_1^{\phi,j} \notin E_\phi.$$

Then we say that $\mathcal{A}$ is equality-pattern respecting.

Yilek claimed in [35] that a further simplification is possible in his Reset Attack (RA) setting, namely that there is a reduction from any adversary making q **LR** queries to an adversary making just one **LR** query. One might hope that a corresponding result would be possible in our Related Randomness setting. Only a sketch proof is given for the RA setting claim in [35], but it appears to be flawed.[3] While we do not have a separation between models with 1 and $q > 1$ **LR** queries for either Yilek's RA or our RRA setting, neither have we been able to prove the desired simplification from q to 1 **LR** query.

[3] The adversary with one **LR** query, which we will call $\mathcal{A}_1$, is supposed to simulate the game for $\mathcal{A}_q$, the adversary with q **LR** queries. Adversary $\mathcal{A}_1$ first guesses an index $i \in \{1, \ldots, q\}$. When adversary $\mathcal{A}_q$ makes its ith **LR** query, $\mathcal{A}_1$ passes the query to its own **LR** oracle. For all other queries, $\mathcal{A}_1$ is supposed to pass either m_0 or m_1 to its **Enc** oracle. However, this simulation is not always possible because of the necessary equality-pattern restrictions in the RA setting. For example, an adversary making two **LR** queries may submit the pairs (m_0, m_1) and (m_1, m_0) to its **LR** oracle, as these are equality-pattern respecting in the model of [35]. Without loss of generality, the simulating adversary $\mathcal{A}_1$ passes the pair (m_0, m_1) to its **LR** oracle, and then, for **LR** query (m_1, m_0), submits either m_0 or m_1 to its **Enc** oracle. However, $\mathcal{A}_1$ would no longer be equality-pattern respecting, even though the original adversary is. Fortunately, the main construction in [35] is still secure against an adversary making multiple **LR** queries.

3.3 Function Restrictions

Above, we briefly alluded to the fact that the class of functions Φ used by our RRA adversaries must be restricted in various ways. The example given showed that constant functions must always be excluded. Here, we exhibit much stronger necessary conditions on Φ that must be satisfied, namely output-unpredictability and collision-resistance. These notions are closely related to notions with the same names arising in the setting of related key security for PRFs that was considered in [5]. Here, however, we are concerned with functions acting on the randomness used in PKE schemes rather than on PRF keys.

Definition 11 (Output-unpredictability for Φ). *Let Φ be a set of functions from* Rnd *to* Rnd*. Let α and β be positive integers. Then the (α, β)-output-unpredictability of Φ is defined to be:*

$$\mathrm{InSec}^{\mathrm{up}}_{\Phi}(\alpha, \beta) = \max_{P \subseteq \Phi, X \subseteq \mathcal{R}, |P| \leq \alpha, |X| \leq \beta} \left\{ \mathbb{P}\left[r \leftarrow_{\$} \mathtt{Rnd} : \{\phi(r) : \phi \in P\} \cap X \neq \emptyset \right] \right\}.$$

Definition 12 (Collision-resistance for Φ). *Let Φ be a set of functions from* Rnd *to* Rnd*. Let α be a positive integer. Then the α-collision-resistance of Φ is defined to be:*

$$\mathrm{InSec}^{\mathrm{cr}}_{\Phi}(\alpha) = \max_{P \subseteq \Phi, |P| \leq \alpha} \left\{ \mathbb{P}\left[r \leftarrow_{\$} \mathtt{Rnd} : |\{\phi(r) : \phi \in P\}| < |P| \right] \right\}.$$

Regarding these two definitions, we have the two following results.

Theorem 1 (Necessity of output-unpredictability). *Let Φ be a class of functions from* Rnd *to* Rnd*. Suppose there are natural numbers $\alpha = \mathrm{poly}_1(\lambda)$ and $\beta = \mathrm{poly}_2(\lambda)$ such that $\mathrm{InSec}^{\mathrm{up}}_{\Phi}(\alpha, \beta) = p$, where $p := p(\lambda)$ is non-negligible. Then no* PKE *scheme can be* RRA-ATK *secure with respect to the class of functions Φ.*

Proof. By definition, there exists a set $P \subseteq \Phi$ and a set $X \subseteq$ Rnd, both of polynomial size, such that

$$\mathbb{P}\left[r \leftarrow_{\$} \mathtt{Rnd} : \{\phi(r) : \phi \in P\} \cap X \neq \emptyset \right] = p.$$

We will construct a polynomial-time adversary $\mathcal{A}$ that has advantage p against any scheme PKE. Let pk^* denote the target public key in the RRA-ATK security game. The adversary, $\mathcal{A}$, chooses two distinct messages m_0 and m_1 and computes $E_0 = \{\mathtt{PKE.E}(pk^*, m_0; r) : r \in X\}$ and $E_1 = \{\mathtt{PKE.E}(pk^*, m_1; r) : r \in X\}$. Then $\mathcal{A}$ requests **LR** oracle outputs for $(m_0, m_1, 1, \phi)$ for all $\phi \in P$. Let $E_\phi = \{\mathtt{PKE.E}(pk^*, m_b; \phi(r_1)) : \phi \in P\}$ denote the set of responses to $\mathcal{A}$'s **LR** queries. If $E_b \cap E_\phi \neq \emptyset$, then $\mathcal{A}$ outputs b, otherwise $\mathcal{A}$ chooses $b \leftarrow_{\$} \{0, 1\}$ and outputs b. Let **pred** denote the event that $E_b \cap E_\phi \neq \emptyset$, then

$$\begin{aligned}
\mathbf{Adv}^{\text{rra-atk}}_{\mathtt{PKE},\mathcal{A}}(\lambda) &= 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\mathtt{PKE}}(\lambda) \Rightarrow 1] - 1 \\
&= 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\mathtt{PKE}}(\lambda) \Rightarrow 1 \mid \mathbf{pred}] \cdot \mathbb{P}[\mathbf{pred}] \\
&\quad + 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\mathtt{PKE}}(\lambda) \Rightarrow 1 \mid \overline{\mathbf{pred}}] \cdot \mathbb{P}[\overline{\mathbf{pred}}] - 1 \\
&= 2\left(p + \frac{1}{2}(1-p)\right) - 1 \\
&= p.
\end{aligned}$$

The third line follows because if **pred** occurs then $\mathcal{A}$ wins with probability 1, whilst if **pred** does not occur then $\mathcal{A}$ guesses and hence wins with probability 1/2. No **Dec** queries are required for this attack, so it applies to both the CPA and CCA settings.

Theorem 2 (Necessity of collision-resistance). *Let Φ be a class of functions from* Rnd *to* Rnd*. Suppose there is a natural number $\alpha = \mathrm{poly}_1(\lambda)$ such that $\mathrm{InSec}^{\mathrm{cr}}_{\Phi}(\alpha) = p$, where $p := p(\lambda)$ is non-negligible. Then no* PKE *scheme can be* RRA-ATK *secure with respect to the class of functions Φ.*

Proof. By definition, there exists a subset $P \subseteq \Phi$ of polynomial size such that

$$\mathbb{P}\left[r \leftarrow_{\$} \mathtt{Rnd} : |\{\phi(r) : \phi \in P\}| < |P| \right] = p.$$

We construct a polynomial-time adversary $\mathcal{A}$ that has advantage $p/2$. Let the target public key be pk^*. Then $\mathcal{A}$ chooses $|P| + 1$ distinct messages $m_0, m_1, \ldots, m_{|P|}$ and assigns an index to each $\phi \in P$. For i from 1 to $|P|$, $\mathcal{A}$ requests **LR** oracle output for query $(m_0, m_i, 1, \phi_i)$. Let the output of the **LR** oracle for the ith query be c_i. Let **coll** denote the event that $\phi_i(r_1) = \phi_j(r_1)$ for some $i \neq j$, where r_1 is the randomness chosen by the game for index 1. Adversary $\mathcal{A}$ outputs $b = 0$ if $c_i = c_j$ for some $i \neq j$. Otherwise, $\mathcal{A}$ chooses $b \leftarrow_{\$} \{0, 1\}$ and outputs b. Then,

Alg. $\texttt{Hash-PKE.K}(1^\lambda)$:	Alg. $\texttt{Hash-PKE.E}(pk, m)$:	Alg. $\texttt{Hash-PKE.D}(sk, c)$:
$(pk, sk) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$	$r \leftarrow_\$ \texttt{Rnd}$	$m \leftarrow \texttt{PKE.D}(sk, m)$
	$c \leftarrow \texttt{PKE.E}(pk, m; H(pk\|\|m\|\|r))$	return m
	return c	

Fig. 10. Scheme Hash-PKE built from a PKE scheme PKE and a hash function H.

$$\begin{aligned}
\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PKE},\mathcal{A}}(\lambda) &= 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\texttt{PKE}}(\lambda) \Rightarrow 1] - 1 \\
&= 2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\texttt{PKE}}(\lambda) \Rightarrow 1 \mid b = 0 \wedge \mathbf{coll}] \cdot \mathbb{P}[b = 0 \wedge \mathbf{coll}] \\
&\quad +2 \cdot \mathbb{P}[\text{RRA-ATK}^{\mathcal{A}}_{\texttt{PKE}}(\lambda) \Rightarrow 1 \mid \overline{b = 0 \wedge \mathbf{coll}}] \cdot \mathbb{P}[\overline{b = 0 \wedge \mathbf{coll}}] - 1 \\
&= 2 \left(\frac{1}{2}p + \frac{1}{2}(1 - \frac{1}{2}p) \right) - 1 \\
&= \frac{1}{2}p.
\end{aligned}$$

This follows because if $b = 0$ and **coll** occurs (which happens with probability $p/2$) then $\mathcal{A}$ wins with probability 1, whilst if **coll** does not occur or $b = 1$ then $\mathcal{A}$ guesses and hence wins with probability $1/2$. No **Dec** queries are required for this attack, so this argument applies equally well to both the CPA and CCA settings.

We note that many classes of functions that arise from practical attacks satisfy the output-unprecdictability and collision-resistance conditions. For example, the class of functions that flip bits at certain positions, or the class of functions that fix the value of certain bits, are both output-unpredictable and collision-resistant (provided at least a polynomial number of bits are not fixed, in the latter case).

4 Construction in the Random Oracle Model

We have seen that the class of functions Φ must be collision-resistant and output-unpredictable in order to be secure against related randomness attacks. In the ROM, these two conditions are in fact also *sufficient* to ensure security in our *strongest* RRA-ATK models, in the following sense: given a hash function H, any PKE scheme PKE that is IND-ATK secure, and a set of functions Φ that is both collision-resistant and output-unpredictable, the scheme Hash-PKE constructed from PKE as in Figure 10 is Φ-RRA-ATK secure in the ROM. The next result formalises this claim.

Theorem 3. *Suppose $\mathcal{A}$ is a Φ-restricted, equality-pattern respecting adversary in the* RRA-ATK *game against the scheme* Hash-PKE *defined in Figure 10. Suppose $\mathcal{A}$ requests encryptions for q_ϕ distinct functions, queries q_r randomness indices, and makes q_{RO} random oracle queries. Then there exists an IND-ATK adversary $\mathcal{C}$ against* PKE *such that:*

$$\mathbf{Adv}^{\text{rra-atk}}_{\texttt{Hash-PKE},\mathcal{A}}(\lambda) \leq q_r \cdot q_{LR} \cdot \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{C}}(\lambda) + 2q_r \cdot \text{InSec}^{\Phi}_{\text{cr}}(q_\phi) + 2q_r \cdot \text{InSec}^{\Phi}_{\text{up}}(q_\phi, q_{RO}).$$

Adversary $\mathcal{C}$'s running time is approximately the same as that of $\mathcal{A}$. In the CCA game, $\mathcal{C}$ makes the same number of decryption queries as $\mathcal{A}$.

Proof. First, we invoke Lemma 1, so that we now only have to prove the theorem for an adversary using just one randomness value, which we assume to be r^*. Without loss of generality, we assume that queries to the random oracle take the form $pk||m||r$, where pk is a public key, m is a message and r is a randomness value. Let **pred** denote the event that $\mathcal{A}$ makes a query $X = pk||m||r$ to the random oracle and this value X is used by the **Enc** or **LR** oracle as an input to the random oracle to encrypt a message m with randomness $\phi(r^*)$ under public key pk. Let **coll** denote the event that $\mathcal{A}$ queries distinct functions ϕ_1 and ϕ_2 such that $\phi_1(r^*) = \phi_2(r^*)$. Then

$$\begin{aligned}
\mathbf{Adv}^{\text{rra-atk}}_{\texttt{Hash-PKE},\mathcal{A}}(\lambda) &= 2 \cdot \mathbb{P}[\text{RRA-RO-ATK}^{\mathcal{A}}_{\texttt{Hash-PKE}} \Rightarrow 1] - 1 \\
&\leq 2 \cdot \left(\mathbb{P}[\text{RRA-RO-ATK}^{\mathcal{A}}_{\texttt{Hash-PKE}} \Rightarrow 1 \mid \overline{\mathbf{coll} \vee \mathbf{pred}}] + \mathbb{P}[\mathbf{coll} \vee \mathbf{pred}] \right) - 1 \\
&\leq 2 \cdot \mathbb{P}[\text{IND-ATK}^{\mathcal{B}}_{\texttt{PKE}} \Rightarrow 1] - 1 + 2 \cdot \mathbb{P}[\mathbf{coll}] + 2 \cdot \mathbb{P}[\mathbf{pred}] \\
&\leq \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{B}}(\lambda) + 2 \cdot \text{InSec}^{\Phi}_{\text{cr}}(q_\phi) + 2 \cdot \text{InSec}^{\Phi}_{\text{up}}(q_\phi, q_{RO}) \\
&\leq q_{LR} \cdot \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{C}}(\lambda) + 2 \cdot \text{InSec}^{\Phi}_{\text{cr}}(q_\phi) + 2 \cdot \text{InSec}^{\Phi}_{\text{up}}(q_\phi, q_{RO}).
\end{aligned}$$

Alg. PRF-PKE.K(1^λ):	Alg. PRF-PKE.E(pk, m):	Alg. PRF-PKE.D(sk, c):
$(pk, sk) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$	$r \leftarrow_\$ \texttt{Rnd}$	$m \leftarrow \texttt{PKE.D}(sk, c)$
	$r' \leftarrow F_r(pk\|\|m)$	return m
	$c \leftarrow \texttt{PKE.E}(pk, m; r')$	
	return c	

Fig. 11. Scheme PRF-PKE built from a standard PKE scheme, PKE and a PRF, F.

The third line follows because if neither **coll** nor **pred** occurs, then the inputs to the random oracle are distinct and unknown, so the outputs may be replaced with random values chosen independently and uniformly at random. Hence, a standard IND-ATK adversary $\mathcal{B}$ can simulate this game for $\mathcal{A}$. For $\mathcal{A}$'s **Enc** queries, $\mathcal{B}$ chooses a fresh random value and uses this to encrypt. For **LR** queries, $\mathcal{B}$ forwards the message pair to his own **LR** oracle. **Dec** queries are forwarded to $\mathcal{B}$'s **Dec** oracle and $\mathcal{B}$ returns a random value for random oracle queries. From $\mathcal{A}$'s perspective, $\mathcal{B}$ provides a perfect simulation. When $\mathcal{A}$ outputs a bit, $\mathcal{B}$ outputs the same bit. The final line comes from a straightforward hybrid argument. $\mathcal{B}$ is allowed multiple **LR** queries, but this may be simulated by a standard IND adversary that is allowed only 1 **LR** query, with a security loss. The simulator $\mathcal{C}$ guesses an index $j \in \{1 \ldots, q_{LR}\}$ and forwards the target public key. For $\mathcal{B}$'s i-th query, if $i < j$ (resp. $i > j$) $\mathcal{C}$ encrypts m_0 (resp. m_1) with fresh randomness and returns the ciphertext to $\mathcal{B}$. If $i = j$, $\mathcal{C}$ forwards (m_0, m_1) to his own **LR** oracle and forwards the output to $\mathcal{B}$. At the end of the simulation $\mathcal{C}$ outputs the same bit as $\mathcal{B}$. This completes the proof.

Bellare *et al.* [2] introduced the *randomized-encrypt-with-hash* (REwH) method to protect against randomness failures in public key encryption. This method amounts to incorporating as much context as possible via hashing when setting up randomness. It was further developed in [29] as a general purpose technique applicable to multiple cryptographic primitives. The construction in Figure 10 can be seen as an instance of this method, in that the "random value" used during encryption is replaced with a hash of the public key, the message to be encrypted, and the actual random value. Theorem 3 then shows that hedging in this way not only protects against the various forms of randomness failure considered in [2, 29], but also protects against failures in our related randomness setting, to the maximum extent possible.

Now that we have shown a necessary and sufficient condition for RRA security in the ROM, the challenge is to extend our results to the standard model. The remainder of this paper is concerned with achieving this goal.

5 Related Randomness Security for PKE from RKA-PRFs

Since the RA setting of [35] is a special case of our RRA setting, an obvious way to try to achieve RRA security is to extend the main construction from [35]. That construction combines a PRF with an IND-ATK secure PKE scheme. Specifically, the randomness r is used as a key to the PRF, and the input to the PRF is the "context" $pk||m$; the output from the PRF is then used as the actual randomness for encryption. This construction extends directly to our setting, and security is guaranteed against Φ-restricted adversaries in our strongest RRA-ATK models, under the assumption that the PRF is Φ-RKA-secure (i.e. secure against related key attacks for the *same* class of functions Φ). Thus the construction transfers RKA security for PRFs to RRA-ATK security for PKE. Figure 11 formalises the construction, and Theorem 4 our security result. Notice that our RO scheme in Section 4 may be interpreted as an instantiation of the scheme in Figure 11, since a random oracle can be viewed as an (unkeyed) RKA-PRF.

Theorem 4. *Suppose $\mathcal{A}$ is a Φ-restricted, equality-pattern respecting adversary in the* RRA-ATK *game against the scheme* PRF-PKE *defined in Figure 11. Suppose $\mathcal{A}$ makes q_{LR} **LR** queries, q_s **Enc** queries, and uses q_r randomness indices. Then there exists a Φ-restricted RKA-PRF adversary $\mathcal{B}$ and an IND-ATK adversary $\mathcal{C}$ such that*

$$\mathbf{Adv}^{\text{rra-atk}}_{\texttt{PRF-PKE},\mathcal{A}}(\lambda) \leq q_{LR} \cdot q_r \cdot \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{C}}(\lambda) + 2q_r \cdot \mathbf{Adv}^{\text{rka-prf}}_{F,\mathcal{B}}(\lambda).$$

*Adversaries $\mathcal{B}$ and $\mathcal{C}$ run in approximately the same time as $\mathcal{A}$. Adversary $\mathcal{C}$ makes 1 **LR** query and the same number of **Dec** queries as $\mathcal{A}$. Adversary $\mathcal{B}$ makes at most $q_{LR} + s$ queries to its oracle.*

Proof. We first apply Lemma 1, so that we may concentrate on an adversary using just one randomness value. Let G_0 be the real RRA-ATK security game played by an adversary $\mathcal{A}$ against the scheme PRF-PKE and let G_1 be the game where outputs of the PRF F are replaced with values chosen uniformly at random. That is, in G_1, each encryption uses fresh random coins rather than using outputs from F. We first claim that there is an adversary $\mathcal{B}$ against the Φ-RKA-PRF security of F such that:

$$\mathbb{P}[G_0^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}^{\text{rka-prf}}_{F,\mathcal{B}}(\lambda)$$

Our construction of Adversary $\mathcal{B}$ is as follows. Adversary $\mathcal{B}$ flips a bit b and generates a key pair (pk^*, sk^*). Then, $\mathcal{B}$ gives pk^* to $\mathcal{A}$ and runs $\mathcal{A}$. When $\mathcal{A}$ submits an **Enc** query $(pk, m, 1, \phi)$, $\mathcal{B}$ sends $(\phi, pk||m)$ to its

oracle, uses the output, r' to encrypt, so that $c \leftarrow \mathtt{PKE.E}(pk, m; r')$, and returns c to $\mathcal{A}$. When $\mathcal{A}$ submits an **LR** query $(m_0, m_1, 1, \phi)$, $\mathcal{B}$ sends $(\phi, pk^*||m_b)$ to its oracle and uses the output, r', to encrypt, setting $c \leftarrow \mathtt{PKE.E}(pk^*, m_b; r')$ and returning c to $\mathcal{A}$. In the CCA game, $\mathcal{B}$ generated sk^* so he may use this to respond to any decryption queries. At the end of the simulation, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs 1 if and only if $b = b'$. If $\mathcal{B}$ is in the real world, he simulates G_0, otherwise he simulates G_1.

Now adversary $\mathcal{A}$'s queries in game G_1 can be simulated by an IND-ATK adversary $\mathcal{C}$ against PKE as follows. The adversary $\mathcal{C}$ must guess an index $j \in \{1, \ldots, q_{LR}\}$. For the i-th **LR** query made by $\mathcal{A}$, when $i < j$ (resp. $i > j$) $\mathcal{C}$ responds with an encryption of m_0 (resp. m_1). For the j-th **LR** query made by $\mathcal{A}$, $\mathcal{C}$ forwards the query to his own **LR** oracle and returns the output. At the end of the simulation $\mathcal{C}$ outputs the same bit as $\mathcal{A}$ playing G_1. A standard hybrid argument shows that:

$$2 \cdot \mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - 1 \leq q_{LR} \cdot \mathbf{Adv}^{\text{ind-atk}}_{F,\mathcal{C}}(\lambda).$$

Hence,

$$\begin{aligned}
\mathbf{Adv}^{\text{rra-atk}}_{\mathtt{PRF\text{-}PKE},\mathcal{A}}(\lambda) &= 2 \cdot \mathbb{P}[G_0^{\mathcal{A}} \Rightarrow 1] - 1 \\
&= 2(\mathbb{P}[G_0^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1]) + 2 \cdot \mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - 1 \\
&\leq 2 \cdot \mathbf{Adv}^{\text{rka-prf}}_{F,\mathcal{B}}(\lambda) + q_{LR} \cdot \mathbf{Adv}^{\text{ind-atk}}_{F,\mathcal{C}}(\lambda).
\end{aligned}$$

Combining with the randomness reduction lemma gives the desired result.

The previous theorem is seductively simple, but currently of limited application because the set of known RKA-secure PRFs is rather sparse. RKA-PRFs were first formalised in 2003 by Bellare and Kohno [5], and some initial (though not fully satisfactory) constructions were given in [5] and [26]. Setting these aside, the only known constructions are due to Bellare and Cash [3]. They gave a first construction for an RKA-PRF (based on the Naor-Reingold PRF) which is provably secure under the DDH assumption for related key functions Φ corresponding to component-wise multiplication on the key-space $(\mathbb{Z}_p^*)^{n+1}$. They also provided a second construction achieving a similar result under the DLIN assumption. A third construction for related key functions Φ corresponding to component-wise *addition* on the key-space $(\mathbb{Z}_p)^n$ was recently withdrawn by the authors of [3].

The limited nature of existing RKA-PRF families forces us to find alternative approaches to achieving security in the RRA setting. The application for RKA-PRFs implied by Theorem 4 also provides yet more motivation for the fundamental problem of constructing RKA-PRFs for richer classes of related key function.

6 Related Randomness PKE from CIS Hash Functions

To address some of the limitations encountered in the previous approach, we show how a PKE scheme secure in the RRA setting can be constructed using correlated-input secure (CIS) hash functions as introduced in [20]. While the currently known instantiations of CIS hash functions only allow us to obtain selective HK-RRA-ATK security, we are able to obtain security for a large class of polynomial functions, as opposed to linear functions to which the previous construction is currently restricted.

In its strongest form, a CIS hash function h (with key k) will yield output $h_k(x)$ which is pseudorandom, even when given the hash value of multiple correlated input values $(h_k(\phi_1(x)), \ldots, h_k(\phi_q(x)))$, where the correlation functions $\phi_1, \ldots, \phi_q$ are maliciously chosen. This type of CIS hash function is closely related to RKA-secure PRFs. In fact, the authors of [20] show that given a CIS hash function h, an RKA-secure *weak* PRF F can be obtained simply by exchanging the role of the key and the input of h:

$$F_K(x) := h_x(K).$$

Recall that weak PRF security does not allow an adversary to choose the function inputs, but instead, the inputs are chosen uniformly at random in the security game.

The authors of [20] furthermore give a concrete construction of a CIS hash function secure for a class of correlation functions consisting of uniform-output[4] polynomials of bounded degree, albeit in a restricted security model where the adversary's function queries are non-adaptive. This then yields a non-adaptive, RKA-secure weak PRF.

Unfortunately, such a PRF this is not sufficient for our purposes. Surprisingly, however, by making a relatively simple modification to the above construction of PRFs from CIS hash functions, it is possible to obtain a primitive similar to an RKA-secure (standard) PRF. More specifically, consider a CIS hash function h and a

[4] A polynomial is said to be a uniform-output polynomial if its output range is equal to its domain i.e. evaluating the polynomial on all values in the domain will again yield the elements of the domain.

proc. Initialise(λ, ℓ):
$(\mathcal{K}, \mathcal{D}, \mathcal{R}, h) \leftarrow \texttt{GenFun}(1^\lambda)$
For $i = 1$ to ℓ
  $k_i \leftarrow_\$ \mathcal{K}$
$x \leftarrow_\$ \mathcal{D}$
$b \leftarrow_\$ \{0,1\}$
$\mathcal{S} \leftarrow \emptyset$
$\texttt{func} \leftarrow \texttt{false}$
$\texttt{chal} \leftarrow \texttt{false}$
return $(\mathcal{K}, \mathcal{D}, \mathcal{R}, h)$

proc. Functions$(\phi_1, \ldots, \phi_q)$:
If $\texttt{func} = \texttt{true}$, return $\bot$
$\texttt{func} \leftarrow \texttt{true}$
return $k_1, \ldots, k_\ell$

proc. Hash(i, j):
If $\texttt{func} = \texttt{false}$
or $\texttt{chal} = \texttt{true}$,
  return $\bot$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(i, j)\}$
return $h_{k_i}(\phi_j(x))$

proc. Chal(i^*, j^*):
If $\texttt{func} = \texttt{false}$
or $\texttt{chal} = \texttt{true}$,
  return $\bot$
If $(i^*, j^*) \in \mathcal{S}$,
  return $\bot$
$y_0 \leftarrow_\$ \mathcal{R}$
$y_1 \leftarrow h_{k_{i^*}}(\phi_{j^*}(x))$
$\texttt{chal} \leftarrow \texttt{true}$
return y_b

proc. Finalise(b'):
If $b = b'$, return 1

Fig. 12. Game ℓ-MK-SCI-PR for a family $\mathcal{H}$ of keyed hash functions defined by GenFun.

Alg. $\texttt{CI-Hash-PKE.K}(1^\lambda)$:
$(pk, sk) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$
$k \leftarrow_\$ \texttt{CI-HASH.K}(1^\lambda)$
$(\hat{pk}, \hat{sk}) \leftarrow (pk||k, sk)$

Alg. $\texttt{CI-Hash-PKE.E}(\hat{pk}, m)$:
$(pk||k) \leftarrow \hat{pk}$
$r \leftarrow_\$ \texttt{Rnd}$
$r' \leftarrow h_k(r)$
$r'' \leftarrow F_{r'}(\hat{pk}||m)$
$c \leftarrow \texttt{PKE.E}(pk, m; r'')$
return c

Alg. $\texttt{CI-Hash-PKE.D}(\hat{sk}, c)$:
$m \leftarrow \texttt{PKE.D}(\hat{sk}, c)$
return m

Fig. 13. Scheme CI-Hash-PKE built from PKE scheme PKE, PRF F, and hash function family $\mathcal{H}$.

standard PRF f. We introduce a public parameter c of F which will correspond to the key for h, and then, instead of using the output of h directly, we use h to derive a key for f. More specifically, we define

$$F_{c,K}(x) := f_{h_c(K)}(x).$$

Whilst not strictly an RKA-secure PRF due to the presence of the public parameter c, this primitive allows adaptively chosen inputs x, while remaining secure under related key attacks. This 'partial' RKA-secure PRF will allow us to obtain HK-RRA-ATK secure encryption schemes for the function families of the underlying CIS hash function h. However, to achieve this, we need to extend the definitions and theorems of [20] to the multi-key setting (reflecting the fact that in the HK-RRA setting, our adversary can interact with multiple public keys).

We formally define *multi-key selective correlated-input pseudorandomness* (MK-SCI-PR) for a family of keyed hash functions via the security game shown in Figure 12. The definition is selective in the sense that the adversary is required to submit the correlation functions before seeing the hash function keys used in the game. As in the definition of related randomness security, we consider Φ-restricted adversaries i.e. adversaries who are restricted to submit correlation functions belonging to a given class of functions Φ.

Definition 13. *A family $\mathcal{H}$ of keyed hash functions is said to be* (Φ, ℓ)-MK-SCI-PR *secure if for all Φ-restricted adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ against $\mathcal{H}$, defined as*

$$\mathbf{Adv}^{\ell\text{-mk-sci-pr}}_{\mathcal{H},\mathcal{A}}(\lambda) := 2 \cdot \mathbb{P}[\ell\text{-MK-SCI-PR}^{\mathcal{A}}_{\mathcal{H}}(\lambda) \Rightarrow 1] - 1,$$

is negligible in the security parameter λ.

Based on an ordinary PKE scheme PKE, a PRF F, and a family of hash functions $\mathcal{H}$, we construct a PKE scheme CI-Hash-PKE as shown in Figure 13. The following theorem establishes the selective ℓ-HK-RRA-ATK security of this scheme based on the IND-ATK security of PKE, the multi-key selective CIS security of $\mathcal{H}$, and the (regular) pseudorandomness of F.

Theorem 5. *Suppose $\mathcal{A}$ is a Φ-restricted, equality pattern respecting adversary in the selective ℓ-HK-RRA-ATK game against the scheme* CI-Hash-PKE *in Figure 13. Suppose $\mathcal{A}$ makes q_{LR} **LR** queries, uses q_r randomness indices, and uses q_ϕ functions in its oracle queries. Then there exists a Φ-restricted, multi-key, selective correlated-input hash adversary $\mathcal{B}$, a* PRF *adversary $\mathcal{C}$ and an* IND-ATK *adversary $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\ell\text{-shk-rra-atk}}_{\texttt{CI-Hash-PKE},\mathcal{A}}(\lambda) \leq 2q_\phi \cdot q_r \cdot \mathbf{Adv}^{\ell\text{-mk-sci-pr}}_{\mathcal{H},\mathcal{B}}(\lambda) + 2q_\phi \cdot q_r \cdot \mathbf{Adv}^{\text{prf}}_{F,\mathcal{C}}(\lambda)$$
$$+\ell \cdot q_{LR} \cdot q_r \cdot \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{D}}(\lambda) + \frac{\ell^2 \cdot q_r}{|\text{HashKeySpace}|}.$$

Adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ run in approximately the same time as $\mathcal{A}$. Adversary $\mathcal{C}$ makes at most q_{LR} queries, and $\mathcal{D}$ makes 1 ***LR*** *query and as many* ***Dec*** *queries as $\mathcal{A}$.*

Proof. First, we invoke Lemma 1, so that we now only have to prove the theorem for an adversary using just one randomness value. We prove the theorem via a sequence of game hops and hybrid arguments. Let G_0 be the real correlated-input hash PKE game, which is the non-adaptive version of the game in Figure 7. The game G_1 is the same except for queries on the target key. Rather than using the hash function, the game picks a uniformly random output for each function and uses this value as a key for the PRF.

$$\begin{aligned}\mathbf{Adv}^{\ell\text{-hk-rra-atk}}_{\texttt{CI-Hash-PKE},\mathcal{A}}(\lambda) &= 2 \cdot \mathbb{P}[G_0^{\mathcal{A}}] - 1 \\ &\leq 2 \cdot (\mathbb{P}[G_0^{\mathcal{A}} \mid \overline{\mathbf{coll}}] + \mathbb{P}[\mathbf{coll}]) - 1 \\ &= 2 \cdot (\mathbb{P}[G_0^{\mathcal{A}} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G_1^{\mathcal{A}} \mid \overline{\mathbf{coll}}]) + 2 \cdot \mathbb{P}[\mathbf{coll}] - 1 \\ &\quad +2 \cdot \mathbb{P}[G_1^{\mathcal{A}} \mid \overline{\mathbf{coll}}].\end{aligned}$$

If there are no collisions in the hash function keys, then the difference between G_0 and G_1 is negligible. This can be stated formally as:

Lemma 2. *The difference between the success probability of $\mathcal{A}$ in games G_0 and G_1 is bounded by the advnatage of a multi-key, CI hash adersary $\mathcal{B}$. That is:*

$$\mathbb{P}[G_0^{\mathcal{A}} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G_1^{\mathcal{A}} \mid \overline{\mathbf{coll}}] \leq \mathbf{Adv}^{\ell\text{-mk-sci-pr}}_{\mathcal{H},\mathcal{B}}(\lambda).$$

Proof. We will prove this via a hybrid argument. Let $G_{0,i}$ denote the game in which, for the target public key, for function $k \leq q_\phi - i$ the output of the hash function is real, whereas for function $k > q_\phi - i$ the output is random, rather than using the hash function. Notice that $G_0 = G_{0,0}$ and $G_1 = G_{0,q_\phi}$, so

$$\begin{aligned}\mathbb{P}[G_0^{\mathcal{A}} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G_1^{\mathcal{A}} \mid \overline{\mathbf{coll}}] &= \sum_{n=0}^{q_\phi - 1} \mathbb{P}[G_{0,i}^{\mathcal{A}} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G_{0,i+1}^{\mathcal{A}} \mid \overline{\mathbf{coll}}] \\ &\leq q_\phi \cdot |\mathbb{P}[G_{0,j^*}^{\mathcal{A}} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G_{0,j^*+1}^{\mathcal{A}} \mid \overline{\mathbf{coll}}]|.\end{aligned}$$

If an adversary can distinguish games j^* and $j^* + 1$ when there are no collisions of hash keys, then we may use this adversary to construct a multi-key CI hash adversary that distinguishes with the same probability. The CI-hash adversary $\mathcal{B}$ will simulate either the game G_{0,j^*} or G_{0,j^*+1} for $\mathcal{A}$.

Adversary $\mathcal{B}$ initiates the CI hash game and is given the description of a hash function. Then, $\mathcal{B}$ sets up the game for $\mathcal{A}$.

Setup
$b \leftarrow_\$ \{0,1\}$
guess an index $j^* \in \{0, \ldots, q_\phi - 1\}$
$\texttt{funchoice} \leftarrow \texttt{false}$; $\texttt{target} \leftarrow \texttt{false}$
Keys $\leftarrow \emptyset$; Functions $\leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$; $(p\hat{k}^*, s\hat{k}^*) \leftarrow \emptyset$

When $\mathcal{B}$ has finished this setup procedure, he forwards the description of the hash function to $\mathcal{A}$. Adversary $\mathcal{B}$ will then answer $\mathcal{A}$'s **Func** and **Target** queries as follows:

Func query $(\phi_1, \ldots, \phi_{q_\phi})$
If $\texttt{functions} = \texttt{true}$, $\mathcal{B}$ returns $\perp$
Otherwise $\mathcal{B}$ forwards the query to his CI challenger and receives $k_1, \ldots, k_\ell$
For $i = 1$ to ℓ, $\mathcal{B}$ generates $(pk_i, sk_i) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$
$\mathcal{B}$ sets $p\hat{k}_i = pk_i || k_i$, and $s\hat{k}_i = sk_i$
$\texttt{functions} \leftarrow \texttt{true}$
$\mathcal{B}$ returns $\{p\hat{k}_i\}$ to $\mathcal{A}$
Target query (j)
If $\texttt{target} = \texttt{true}$, $\mathcal{B}$ returns $\perp$
otherwise $\mathcal{B}$ sets $(p\hat{k}^*, s\hat{k}^*) \leftarrow (p\hat{k}_j, s\hat{k}_j)$
$\texttt{target} \leftarrow \texttt{true}$
For $i \neq j$, $\mathcal{B}$ returns $\{s\hat{k}_i\}$

Adversary $\mathcal{A}$ is denied access to the other oracles until `target=true` and `functions=true`. When `target=true` and `functions=true`, $\mathcal{B}$ submits (i, κ) to his CI Hash oracle for all $(i, \kappa) \neq (j, j^*)$ Then $\mathcal{B}$ submits (j, j^*) to his CI challenge oracle. $\mathcal{B}$ keeps a table of inputs and outputs to the CI oracle. $\mathcal{B}$'s challenge oracle will return a value r, which is either the real output for $h_{k_j}(\phi_{q_\phi - j^*}(r))$ or a uniformly random value. This value r should

be stored alongside $(k_j, q_\phi - j^*)$ in the table. For input pairs (k_j, κ), where $\kappa > q_\phi - j^*$, $\mathcal{B}$ replaces the oracle outputs in the table with uniformly random values. (N.B. Adversary $\mathcal{B}$ could (and would in practice) generate this table 'on-the-fly' in response to $\mathcal{A}$'s **Enc** and **LR** queries. However, to keep our presentation clear and uncluttered, we adopt the current approach of generating the whole table before answering $\mathcal{A}$'s **Enc** or **LR** queries.)

Enc query $(\hat{pk}, m, 1, \kappa)$
If $\texttt{target} = \texttt{false}$ or $\texttt{functions} = \texttt{false}$, return $\perp$
$\mathcal{B}$ parses $\hat{pk} = pk_i || k_i$
$\mathcal{B}$ finds $(k_i, \kappa, r_{i,\kappa})$ in the look up table and returns $\texttt{PKE.E}(pk, m; F_{r_{i,\kappa}}(\hat{pk}||m))$
LR query $(m_0, m_1, 1, \kappa)$
If $\texttt{target} = \texttt{false}$ or $\texttt{functions} = \texttt{false}$, return $\perp$
$\mathcal{B}$ finds (k_j, κ) in the table and gets corresponding value $r_{j,\kappa}$
$\mathcal{B}$ returns $\texttt{PKE.E}(pk^*, m_b; F_{r_{j,\kappa}}(\hat{pk}^*||m_b))$
$\mathcal{B}$ adds the ciphertext to the set $\mathcal{S}$
Dec query (c)
If $\texttt{target} = \texttt{false}$ or $\texttt{functions} = \texttt{false}$, return $\perp$
If c is not in $\mathcal{S}$, $\mathcal{B}$ returns $\texttt{PKE.D}(sk_j, c)$
Else $\mathcal{B}$ returns $\perp$

At the end of the simulation, $\mathcal{B}$ outputs 1 if $\mathcal{A}$ outputs $b = b'$. If $\mathcal{B}$ is in the real world, he simulates G_{0,j^*} perfectly. In the random world, he simulates G_{0,j^*+1} perfectly. Hence, we may conclude that

$$|\mathbb{P}[G^{\mathcal{A}}_{0,j^*} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_{j^*+1} \mid \overline{\mathbf{coll}}]| \leq \mathbf{Adv}^{\ell\text{-mk-sci-pr}}_{\mathcal{H},\mathcal{B}}(\lambda).$$

Game 2 is the same as G_1, except that, for the target public key, a fresh output is chosen for each encryption rather than using the PRF. If an adversary can distinguish games 1 and 2 when there are no collisions in the hash keys, then we may use this adversary to win the PRF game.

Lemma 3. *The difference between the success probabilities of any adversary $\mathcal{A}$ is bounded by a PRF adversary $\mathcal{B}$ as follows:*

$$\mathbb{P}[G^{\mathcal{A}}_1 \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_2 \mid \overline{\mathbf{coll}}] \leq q_\phi \cdot \mathbf{Adv}^{\text{prf}}_{F,\mathcal{C}}(\lambda).$$

Proof. Let $G_{1,i}$ denote the game in which, for function $k \leq i$, a uniformly random value is chosen rather than using the PRF, whereas for $k > i$ the PRF is used. Observe that $G_1 = G_{1,0}$ and $G_2 = G_{1,q_\phi}$, from which we see that

$$\begin{aligned}\mathbb{P}[G^{\mathcal{A}}_1 \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_2 \mid \overline{\mathbf{coll}}] &= \sum_{n=0}^{q_\phi - 1} \mathbb{P}[G^{\mathcal{A}}_{1,i} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_{1,i+1} \mid \overline{\mathbf{coll}}] \\ &\leq q_\phi \cdot |\mathbb{P}[G^{\mathcal{A}}_{1,j^*} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_{1,j^*+1} \mid \overline{\mathbf{coll}}]|.\end{aligned}$$

If an adversary can distinguish games G_{1,j^*} and G_{1,j^*+1}, then we may use this adversary to construct a PRF adversary with the same advantage in the PRF game. The PRF adversary $\mathcal{C}$ will simulate either game G_{1,j^*} or G_{1,j^*+1}. Adversary $\mathcal{C}$'s setup procedure for the simulation is as follows:

Setup
$b \leftarrow_\$ \{0, 1\}$
choose an index $j^* \in \{0, \ldots, q_\phi - 1\}$
$r \leftarrow_\$ \texttt{Rnd}$
choose $q_\phi - j^* - 1$ uniformly random PRF keys $\rho_{j^*+2}, \ldots, \rho_{q_\phi}$
$\texttt{funchoice} \leftarrow \texttt{false}$; $\texttt{target} \leftarrow \texttt{false}$
Keys $\leftarrow \emptyset$; Functions $\leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$; $(\hat{pk^*}, \hat{sk^*}) \leftarrow \emptyset$

When $\mathcal{C}$ finishes this setup procedure, he forwards the description of the hash function to $\mathcal{A}$. Then $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

Func query $(\phi_1, \ldots, \phi_{q_\phi})$
If $\texttt{functions} = \texttt{true}$, $\mathcal{C}$ returns $\perp$
otherwise $\mathcal{C}$ chooses hash keys $k_1, \ldots, k_\ell$ uniformly at random, making sure they are all distinct
For $i = 1$ to ℓ, $\mathcal{C}$ generates $(pk_i, sk_i) \leftarrow_\$ \texttt{PKE.K}(1^\lambda)$
$\mathcal{C}$ sets $\hat{pk}_i = pk_i || k_i$ and $\hat{sk}_i = sk_i$
$\texttt{functions} \leftarrow \texttt{true}$
$\mathcal{C}$ returns $\{\hat{pk}_i\}$ to $\mathcal{A}$

Target query (j)
If $\texttt{target} \leftarrow \texttt{true}$, $\mathcal{C}$ returns $\perp$
otherwise $\mathcal{C}$ sets $(\hat{pk}^*, \hat{sk}^*) \leftarrow (\hat{pk}_j, \hat{sk}_j)$
$\texttt{target} \leftarrow \texttt{true}$
For $i \neq j$, $\mathcal{C}$ returns $\{\hat{sk}_i\}$
Enc query $(\hat{pk}, m, 1, \kappa)$
$\mathcal{C}$ parses $\hat{pk} = pk_i||k_i$
if $i = j$,
if $\kappa < j^* + 1$, $\mathcal{C}$ chooses $r' \leftarrow_{\$} \texttt{Rnd}$
else if $\kappa = j^* + 1$, $\mathcal{C}$ submits $\hat{pk}_i||m$ to his oracle and recieves output r'
else if $\kappa > j^* + 1$, $\mathcal{C}$ computes $r' \leftarrow F_{\rho_\kappa}(\hat{pk}||m)$
else $\mathcal{C}$ computes $r' \leftarrow F_{h_{k_i}(\phi_\kappa(r))}(m)$
$\mathcal{C}$ returns $c \leftarrow \texttt{PKE.E}(pk_i, m; r')$
LR query $(m_0, m_1, 1, \kappa)$
if $\kappa < j^* + 1$, $\mathcal{C}$ chooses $r' \leftarrow_{\$} \texttt{Rnd}$
else if $\kappa = j^* + 1$, $\mathcal{C}$ submits $\hat{pk}_j||m$ to his oracle and recieves output r'
else if $\kappa > j^* + 1$, $\mathcal{C}$ computes $r' \leftarrow F_{\rho_\kappa}(\hat{pk}_j||m_b)$
$\mathcal{C}$ returns $\texttt{PKE.E}(pk^*, m_b; r')$ $\mathcal{C}$ adds the ciphertext c to $\mathcal{S}$
Dec query (c)
If c is not in $\mathcal{S}$, $\mathcal{C}$ returns $\texttt{PKE.D}(sk_j, c)$.
Else $\mathcal{C}$ returns $\perp$

At the end of the simulation, $\mathcal{C}$ outputs 1 if and only if $\mathcal{A}$ outputs $b = b'$. If $\mathcal{C}$ is in the real world, he simulates G_{1,j^*} perfectly. In the random world, he simulates G_{1,j^*+1} perfectly. Hence, we may conclude that

$$|\mathbb{P}[G^{\mathcal{A}}_{1,j^*} \mid \overline{\mathbf{coll}}] - \mathbb{P}[G^{\mathcal{A}}_{1,j^*+1} \mid \overline{\mathbf{coll}}]| \leq \mathbf{Adv}^{\mathrm{prf}}_{F,\mathcal{C}}(\lambda).$$

Finally, the game G_2 may be simulated by a standard IND-ATK adversary. That is:

Lemma 4. *For any adversary $\mathcal{A}$, we may bound $\mathcal{A}$'s advantage by an IND-ATK adversary's advantage as follows:*

$$2 \cdot \mathbb{P}(G^{\mathcal{A}}_2 \mid \overline{\mathbf{coll}}) - 1 \leq \ell \cdot q_{LR} \cdot \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{D}}(\lambda).$$

Proof. Again, we use a hybrid argument. Let $G_{2,i}$ denote the game in which, for the kth **LR** query, if $k \leq q_{LR} - i$, **LR** queries are answered with an encryption of m_0, whilst if $k > q_{LR} - i$ **LR** queries are answered with an encryption of m_1.

$$\begin{aligned} 2 \cdot \mathbb{P}(G^{\mathcal{A}}_2 \mid \overline{\mathbf{coll}}) - 1 &= |\mathbb{P}(\mathcal{A}^{G_{2,0}} \Rightarrow 1 \mid \overline{\mathbf{coll}}) - \mathbb{P}(\mathcal{A}^{G_{2,q_{LR}}} \Rightarrow 1 \mid \overline{\mathbf{coll}})| \\ &\leq q_{LR} \cdot |\mathbb{P}(\mathcal{A}^{G_{2,j^*}} \Rightarrow 1 \mid \overline{\mathbf{coll}}) - \mathbb{P}(\mathcal{A}^{G_{2,j^*+1}} \Rightarrow 1 \mid \overline{\mathbf{coll}})|. \end{aligned}$$

If an adversary $\mathcal{A}$ can distinguish the games G_{2,j^*} and G_{2,j^*+1}, then we may construct a standard IND-ATK adversary that distinguishes with the same advantage.

The IND-ATK adversary $\mathcal{D}$ first runs his **Initialise** procedure and is given a public key pk^*. Then $\mathcal{D}$ sets-up the simulation for $\mathcal{A}$ as follows:

Setup
guesses an index $j^* \in \{0, \ldots, q_{LR} - 1\}$
choose a uniformly random $t \in \{1, \ldots, \ell\}$
choose $r \leftarrow_{\$} \texttt{Rnd}$
ctr $\leftarrow 1$
$\texttt{funchoice} \leftarrow \texttt{false}$; $\texttt{target} \leftarrow \texttt{false}$
Keys $\leftarrow \emptyset$; Functions $\leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$; $(\hat{pk}^*, \hat{sk}^*) \leftarrow \emptyset$

When completed, the IND-ATK adversary $\mathcal{D}$ forwards the description of the hash function to $\mathcal{A}$ and answers $\mathcal{A}$'s oracle queries as follows:

Func query $(\phi_1, \ldots, \phi_{q_\phi})$
If $\texttt{functions} = \texttt{true}$, $\mathcal{D}$ returns $\perp$
otherwise $\mathcal{D}$ chooses hash keys $k_1, \ldots, k_\ell$ uniformly at random, making sure they are all distinct
For $i \in \{1, \ldots, t-1, t+1, \ldots, \ell\}$ $\mathcal{D}$ generates $(pk_i, sk_i) \leftarrow_{\$} \texttt{PKE.K}(1^\lambda)$
$\mathcal{D}$ sets $pk_t = pk^*$
For $i = 1$ to ℓ, $\mathcal{D}$ sets $\hat{pk}_i = pk_i||k_i$ and $\hat{sk}_i = sk_i$
$\texttt{functions} \leftarrow \texttt{true}$
$\mathcal{D}$ returns $\{\hat{pk}_i\}$ to $\mathcal{A}$

Target query (j)
If $\texttt{target} = \texttt{true}$, $\mathcal{D}$ returns $\perp$
$\mathcal{D}$ sets $(\hat{pk}^*, \hat{sk}^*) \leftarrow (\hat{pk}_j, \hat{sk}_j)$
$\texttt{target} \leftarrow \texttt{true}$
For $i \neq j$, $\mathcal{D}$ returns $\{\hat{sk}_i\}$
Enc query $(\hat{pk}, m, 1, \kappa)$
$\mathcal{D}$ parses $\hat{pk} = pk_i || k_i$
if $i = j$, $\mathcal{D}$ chooses $r' \leftarrow_{\$} \texttt{Rnd}$
else $\mathcal{D}$ computes $r' \leftarrow F_{h_{k_i}(\phi_\kappa(r))}(m)$
$\mathcal{D}$ returns $c \leftarrow \texttt{PKE.E}(pk_i, m; r')$
LR query $(m_0, m_1, 1, \kappa)$
if ctr $= q_{LR} - j^*$, $\mathcal{D}$ submits (m_0, m_1) to his own **LR** oracle and receives c
else, $\mathcal{D}$ chooses $r' \leftarrow_{\$} \texttt{Rnd}$
if ctr $< q_{LR} - j^*$, $\mathcal{D}$ computes $c \leftarrow \texttt{PKE.E}(pk_j, m_0; r')$
else if ctr $> q_{LR} - j^*$, $\mathcal{D}$ computes $c \leftarrow \texttt{PKE.E}(pk_j, m_1; r')$
ctr $\leftarrow$ ctr $+ 1$
$\mathcal{D}$ adds the cipherext c to the set $\mathcal{S}$
$\mathcal{D}$ returns c
Dec query (c)
If c is not in $\mathcal{S}$, $\mathcal{D}$ returns $\texttt{PKE.D}(sk_j, c)$.
Else $\mathcal{D}$ returns $\perp$

At the end of the simulation, $\mathcal{D}$ outputs the same bit as $\mathcal{A}$. If $\mathcal{A}$ has chosen pk_t as his target public key (which he does with probability $1/\ell$) then, when $\mathcal{D}$ is given an encryption of m_0, he simulates G_{2,j^*} perfectly. If he receives an encryption of m_1, he simulates G_{2,j^*+1} perfectly. Hence, we may conclude that

$$\frac{1}{\ell}|\mathbb{P}(\mathcal{A}^{G_{2,j^*}} \Rightarrow 1 \mid \overline{\textbf{coll}}) - \mathbb{P}(\mathcal{A}^{G_{2,j^*+1}} \Rightarrow 1 \mid \overline{\textbf{coll}})| \leq \mathbf{Adv}^{\text{ind-atk}}_{\texttt{PKE},\mathcal{D}}(\lambda).$$

The theorem follows by combining the preceding lemmas.

It remains to show that we can instantiate a hash function satisfying the above defined multi-key correlated-input security notion. We achieve this by extending the security results for the CIS hash function defined in [20]. Concretely, the CIS hash function from [20] is defined as follows:

$\texttt{GenFun}(1^\lambda)$: Pick a group $\mathbb{G}$ of prime order p, and set the keyspace to $\mathcal{K} = \mathbb{G} \times \mathbb{Z}_p$, the domain to $\mathcal{D} = \mathbb{Z}_p$, and the range to $\mathcal{R} = \mathbb{G}$. Return $(\mathcal{K}, \mathcal{D}, \mathcal{R}, h)$ where h is a description of the function defined below.
$h_k(x)$: For $k \in \mathcal{K}$ and $x \in \mathcal{D}$, parse k as $(g, a) \in \mathbb{G} \times \mathbb{Z}_p$ and return

$$h_k(x) = g^{\frac{1}{x+a}},$$

where $1/(x + a)$ is computed modulo p.

Based on the decisional q-Diffie Hellman Inversion (q-DDHI) assumption in $\mathbb{G}$ (see Appendix 2.5), and extending the results of [20], we are able to show that the above hash function achieves multi-key correlated-input pseudorandomness for a class of functions consisting of uniform-output polynomials of bounded degree.

Theorem 6. *Assume the decisional q-DDHI assumption holds in $\mathbb{G}$, and let Φ be a class of uniform-output polynomials over $\mathbb{Z}_p$. Then there exists no polynomial-time Φ-restricted adversary $\mathcal{A}$ with non-negligible advantage in the (Φ, ℓ)-MK-SCI-PR security game when interacting with $\mathcal{H}$ defined as above, provided that $\ell \cdot d \leq q + 1$, where d is an upper bound on the sum of the degrees of the polynomials submitted by $\mathcal{A}$. More precisely, if $\ell \cdot d \leq q + 1$, then for any polynomial-time Φ-restricted $\mathcal{A}$, there exists a polynomial-time algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\ell\text{-mk-sci-pr}}_{\mathcal{H},\mathcal{A}}(\lambda) \leq 2n\ell \cdot \mathbf{Adv}^{q\text{-ddhi}}_{\mathbb{G},\mathcal{B}}(\lambda)$$

where n is the number of polynomials submitted by $\mathcal{A}$.

Proof. Assume that a polynomial-time Φ-respecting adversary $\mathcal{A}$ against the ℓ-MK-SCI-PR security of the family of keyed hash functions $\mathcal{H}$ from Section 6 is given. Using $\mathcal{A}$, we will construct a polynomial-time Φ-respecting adversary $\mathcal{B}$ against the 1-MK-SCI-PR of $\mathcal{H}$ i.e. $\mathcal{B}$ will interact in the 1-MK-SCI-PR game while simulating the ℓ-MK-SCI-PR for $\mathcal{A}$. $\mathcal{B}$ is constructed as follows.

Initially, $\mathcal{B}$ is given a description $(\mathcal{K}, \mathcal{D}, \mathcal{R}, h)$ where $\mathcal{K} = \mathbb{G} \times \mathbb{Z}_p$, $\mathcal{D} = \mathbb{Z}_p$ and $\mathcal{R} = \mathbb{G}$. $\mathcal{B}$ runs $\mathcal{A}$ with input $(\mathcal{K}, \mathcal{D}, \mathcal{R}, h)$ and responds to $\mathcal{A}$'s queries as follows:

Functions$(\phi_1, \ldots, \phi_n)$: $\mathcal{B}$ picks random $b_i \leftarrow_\$ \mathbb{Z}_p$ for $i \in [\ell]$ and computes polynomials $\phi'_{(i-1)n+j} = \phi_j + b_i$ for $i \in [\ell], j \in [n]$. Note that the polynomials $\{\phi'_k\}_{k\in[n\ell]}$ are uniform-output polynomials assuming $\{\phi_i\}_{i\in[n]}$ are. $\mathcal{B}$ submits $\{\phi'_k\}_{k\in[n\ell]}$ as his own **Functions** query.
Upon reception of a hash key $k' = (g', a') \in \mathbb{G} \times \mathbb{Z}_p$, $\mathcal{B}$ picks $x_i \leftarrow_\$ \mathbb{Z}_p$ for $i \in [\ell]$ and computes $k_i = (g_i, a_i) \leftarrow (g^{x_i}, a' + b_i)$. Note that $\{k_i\}_{i\in[\ell]}$ are distributed uniformly in $\mathbb{G} \times \mathbb{Z}_p$. Lastly, $\mathcal{B}$ returns $\{k_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

Hash(i, j) : $\mathcal{B}$ simply submits $k \leftarrow (i-1)n + j$ to his own **Hash** oracle, obtains a value y, and returns y^{x_i} to $\mathcal{A}$. Note that

$$y^{x_i} = \left((g')^{\frac{1}{\phi'_k(x)+a'}}\right)^{x_i} = ((g')^{x_i})^{\frac{1}{\phi_j(x)+a'+b_i}} = g_i^{\frac{1}{\phi_j(x)+a_i}}$$

where x is chosen by the 1-MK-SCI-PR game $\mathcal{B}$ interacts with.

Chal(i^*, j^*) $\mathcal{B}$ simply responds to this query as in the **Hash** query above. Note that if the value y obtained by $\mathcal{B}$ is uniform in $\mathbb{G}$, then so is $\mathcal{B}$'s response to $\mathcal{A}$.

Finalise(b') $\mathcal{B}$ simply submits b' in his own **Finalise** query.

It should be clear from the description above that $\mathcal{B}$'s simulation for $\mathcal{A}$ is perfect, and that whenever $\mathcal{A}$ wins the ℓ-MK-SCI-PR game, $\mathcal{B}$ wins the 1-MK-SCI-PR game.

Taking into account the number of polynomials $\mathcal{B}$ submits in the 1-MK-SCI-PR game and applying the above theorem from [20] completes the proof of Theorem 6.

Note 1. Our 'partial' RKA-secure PRF is only secure when an adversary's function queries are non-adaptive, which is why we are only able to prove selective HK-RRA-ATK security. If we had a result similar to Theorem 6 for adaptive function queries, then we would immediately obtain a PKE scheme that is (adaptively) HK-RRA-ATK secure.

Note 2. The above construction is only shown to achieve HK-RRA-ATK security, as opposed to RRA-ATK security. The technical reason for this is that public keys include a hash key, and the CIS hash function is only assumed to be secure for honestly generated keys. An alternative solution would be to introduce a *common reference string* (CRS) containing a single hash key, and let all users make use of this. While this requires a trusted third party to initially set up the CRS, it would be possible to show RRA-ATK security of the above construction in a security model appropriately extended to model the presence of a CRS.

Likewise, if we had a multi-key CIS hash function that remained secure for maliciously chosen keys, then we would be able to obtain full RRA-ATK security for the above construction. Unfortunately, we are currently unaware of how to obtain such CIS hash functions.

7 Function-Vector Related Randomness Security

Our previous standard model constructions concerned functions ϕ that are linear (scheme `PRF-PKE` analysed in Theorem 4 combined with known RKA-PRF families), or of bounded degree and having unpredictable outputs (scheme `CI-Hash-PKE` analysed in Theorem 6). We now turn our attention to alternative classes of functions. Specifically, we will propose a construction for a PKE scheme that is $\boldsymbol{\Phi}$-FV-RRA-ATK secure for the set $\boldsymbol{\Phi}$ of vectors of functions that are hard to invert, in a sense that we make precise next.

Definition 14. *Let $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$ denote a vector of functions on a set $\mathtt{Rnd}_\lambda$, where $q := q(\lambda)$ is polynomial in the security parameter λ. Let $\delta(\lambda)$ be a function. We say that $\boldsymbol{\phi}$ is* $\delta(\lambda)$-hard-to-invert *if, for all polynomial-time algorithms $\mathcal{A}$ and all sufficiently large λ, we have:*

$$\mathbb{P}[r \leftarrow \mathcal{A}(\phi_1(r), \ldots, \phi_q(r)) : r \leftarrow_\$ \mathtt{Rnd}_\lambda] \leq \delta(\lambda).$$

We say that a set of vectors of functions $\boldsymbol{\Phi}$ is δ-hard-to-invert if each vector $\boldsymbol{\phi} \in \boldsymbol{\Phi}$ is δ-hard-to-invert (note that the vectors in such a set $\boldsymbol{\Phi}$ need not all be of the same dimension, but we assume they each have dimension that is polynomial in λ).

We will now construct a PKE scheme that offers $\boldsymbol{\Phi}$-FV-RRA-CPA security, where $\boldsymbol{\Phi}$ is the set of *all* sufficiently hard-to-invert vectors of functions on the scheme's randomness space `Rnd`. As noted in Section 3, security in this setting is quantified over *all* vectors in $\boldsymbol{\Phi}$, and the adversary is allowed to work with any set of public keys (even maliciously generated) in its attack. This makes our result relatively strong.

To ease the security analysis of our scheme, we will use a variant of the standard DDH assumption.

Definition 15. *Let $\mathbb{G}$ be a cyclic group of prime order p. The game q-DDH in $\mathbb{G}$ selects generators $g_1, \ldots, g_q$ from $\mathbb{G}$ and a bit $b \leftarrow_\$ \{0,1\}$. The game chooses $(r_1, \ldots, r_q) \leftarrow_\$ \mathbb{Z}_p^q$ and $r \leftarrow_\$ \mathbb{Z}_p$. If $b = 1$, the game returns $g_1, \ldots, g_q, g_1^r, \ldots, g_q^r$ to the adversary. Otherwise, the game returns $g_1, \ldots, g_q, g_1^{r_1}, \ldots, g_q^{r_q}$ to the adversary. When the adversary returns a bit b', the game outputs 1 if and only if $b = b'$. We then define the advantage of a q-DDH adversary $\mathcal{A}$ to be:*

$$\mathbf{Adv}^{q\text{-ddh}}_{\mathbb{G},\mathcal{A}}(\lambda) = 2 \cdot \mathbb{P}[q\text{-DDH}^{\mathcal{A}}_{\mathbb{G}}(\lambda) \Rightarrow 1] - 1.$$

Alg. $\texttt{mBHHO.K}(1^\lambda)$:	Alg. $\texttt{mBHHO.E}(pk, m)$:	Alg. $\texttt{mBHHO.D}(sk, (c_1, c_2))$:
$g_1, \ldots, g_k \leftarrow_\$ \mathbb{G}$	$r \leftarrow_\$ \{0,1\}^k$	$(K, r') \leftarrow f(c_1^x)$
$x \leftarrow_\$ \mathbb{Z}_p$	$c_1 = \prod_{i=1}^k g_i^{r_i}$	$m \leftarrow \texttt{DEM.D}(K, c_2)$
$pk = (g_1, \ldots, g_k, g_1^x \ldots, g_k^x)$	$(K, r') \leftarrow f(\prod_{i=1}^k (g_i^x)^{r_i})$	return m
$sk = x$	$r'' \leftarrow F_{r'}(pk \| m)$	
	$c_2 = \texttt{DEM.E}(K, m; r'')$	
	return (c_1, c_2)	

Fig. 14. Modified BHHO scheme $\texttt{mBHHO}$, constructed using a PRF, F, a KDF, f, and a DEM $\texttt{DEM}$.

Assumption 1 (The q-Decisional Diffie Hellman (q-DDH) Assumption) *For any polynomial-time adversary $\mathcal{A}$, and any q that is polynomial in λ, we have:*

$$\mathbf{Adv}^{q\text{-ddh}}_{\mathbb{G},\mathcal{A}}(\lambda) \leq \text{negl}(\lambda).$$

The q-DDH assumption follows from the standard Decisional Diffie Hellman assumption [28].

With these definitions in hand, Figure 14 defines our PKE scheme $\texttt{mBHHO}$ which offers security in the FV-RRA-CPA setting. This scheme is obtained by modifying a PKE scheme of Boneh *et al.* [12] (the $\texttt{BHHO}$ scheme) which Dodis *et al.* [14] showed to be secure in the auxiliary input setting. The scheme makes use of a KDF f and a PRF F with certain domains and ranges, and a DEM $\texttt{DEM}$. To arrive at our modified scheme $\texttt{mBHHO}$, we swap the roles of secret key and randomness in the original $\texttt{BHHO}$ scheme. This then enables us to provide the values $\phi_i(r)$ as auxiliary inputs without undermining the usual IND-CPA security of the scheme; in turn, these values enables our security reduction to properly handle **Enc** queries involving any function ϕ_i. For technical reasons discussed below, we also need to set the randomness space of the scheme to be $\{0,1\}^k$ where $k := k(\lambda)$ denotes a polynomial function of λ. The following theorem gives our formal result concerning the FV-RRA-CPA security of this scheme.

Theorem 7. *Let $\boldsymbol{\Phi}$ be the set of δ-hard-to-invert vectors of functions on $\{0,1\}^k$. Consider any polynomial-size vector of functions $\boldsymbol{\phi} \in \boldsymbol{\Phi}$ and any equality-pattern respecting, $\boldsymbol{\phi}$-FV-RRA-CPA adversary $\mathcal{A}$ against $\texttt{mBHHO}$. Suppose $\mathcal{A}$ makes q_{LR} **LR** queries and uses q_r randomness indices. Then there exists a k-DDH adversary $\mathcal{B}$, a KDF adversary $\mathcal{D}$, a PRF adversary $\mathcal{E}$, and an IND-CPA adversary $\mathcal{F}$, all running in polynomial time, such that:*

$$\begin{aligned}\mathbf{Adv}^{\boldsymbol{\phi}\text{-fv-rra-cpa}}_{\texttt{mBHHO},\mathcal{A}}(\lambda) < {} & 2q_r \cdot \mathbf{Adv}^{k\text{-ddh}}_{\mathbb{G},\mathcal{B}}(\lambda) + 2q_r \cdot \mathbf{Adv}^{\text{kdf}}_{f,\mathcal{D}}(\lambda) \\ & + 2q_r \cdot \mathbf{Adv}^{\text{prf}}_{F,\mathcal{E}}(\lambda) + q_r \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\texttt{DEM},\mathcal{F}}(\lambda) \\ & + 2q_r \sqrt[3]{512\delta k p^4}.\end{aligned}$$

In particular, when δ is sufficiently small, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

The bound in the above theorem is slightly better than that appearing in the PKC 2014 version of this paper because of a small error in our original probability analysis. We have also set the randomness space in the scheme to be $\{0,1\}^{k(\lambda)}$, rather than $\{0,1\}^\lambda$ as in the original presentation. This is important in being able to set parameters so as to achieve a meaningful (i.e. small) value for the term $2q_r \sqrt[3]{512\delta k p^4}$ in our security bound.

Proof. In what follows, we let $r_{1,i}$ denote the ith bit of r_1 and, without loss of generality, we assume that the function of vectors is of size $q = \text{poly}(\lambda)$. First, we invoke Lemma 1, so that we now only have to prove the theorem for an adversary using just one randomness value, which we will call r_1. The proof then uses a sequence of games, as follows:

$\mathbf{G_0}$: G_0 is the real game with the scheme defined in Figure 14.

$\mathbf{G_1}$: G_1 is the same as G_0, except the target public key components $g_1^x, \ldots, g_k^x$ are replaced with $g^{u_1}, \ldots, g^{u_k}$ where g is a group generator and $u_i \leftarrow_\$ \mathbb{Z}_p$. If $\mathcal{A}$'s success probability is significantly different in games G_0 and G_1, then we can use $\mathcal{A}$ to build an adversary $\mathcal{B}$ winning the k-DDH game.

$\mathbf{G_2}$: G_2 is the same as G_1, except for the challenge ciphertexts, which use g^w as the input to the KDF where $w \leftarrow_\$ \mathbb{Z}_p$, rather than using $\prod_{i=1}^k (g_i^{u_i})^{r_{1,i}}$. If $\mathcal{A}$'s success probability is significantly different in games G_1 and G_2, then we can use $\mathcal{A}$ to build an adversary $\mathcal{C}$ that inverts the vector of functions $(\phi_1, \ldots, \phi_q)$.

$\mathbf{G_3}$: G_3 is the same as G_2, except that the output of the KDF for the challenge ciphertexts is replaced by a uniformly random value. If $\mathcal{A}$'s success probability is significantly different in games G_2 and G_3, then we can use $\mathcal{A}$ to build an adversary $\mathcal{D}$ that wins the KDF security game.

$\mathbf{G_4}$: G_4 is the same as G_3, except that the PRF outputs used in constructing the challenge ciphertexts are replaced by uniformly random values. If $\mathcal{A}$'s success probability is significantly different in games G_3 and G_4, then we can use $\mathcal{A}$ to build an adversary $\mathcal{E}$ that wins the PRF security game. Finally, $\mathcal{A}$'s success in G_4 can be related to that of an IND-CPA adversary $\mathcal{F}$ against the DEM component of the scheme.

We now analyse each of the game transitions in more detail.
G_0 – G_1: We will prove the following:

Lemma 5. *For any adversary $\mathcal{A}$, there exists a λ-DDH adversary $\mathcal{B}$ such that:*

$$\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_0^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}^{\lambda\text{-ddh}}_{\mathbb{G},\mathcal{B}}(\lambda).$$

Proof. The k-DDH adversary $\mathcal{B}$ with access to $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$ will simulate either G_0 or G_1 for $\mathcal{A}$. Adversary $\mathcal{B}$ is given $g_1, \ldots, g_k, g'_1, \ldots, g'_k$, where $g'_1, \ldots, g'_k$ is either $g_1^x, \ldots, g_k^x$ or $g^{u_1}, \ldots, g^{u_k}$ for uniformly random u_i. Adversary $\mathcal{B}$ sets $b \leftarrow_\$ \{0,1\}$ and $r_1 \leftarrow_\$ \{0,1\}^k$. It then simulates **proc. Initialise** in the $\boldsymbol{\phi}$-FV-RRA-CPA security game by forwarding $pk^* = (g_1, \ldots, g_k, g'_1, \ldots, g'_k)$ to $\mathcal{A}$. Then $\mathcal{B}$ answers $\mathcal{A}$'s queries as follows:

Enc query (pk, m, i)
  $\mathcal{B}$ returns $\mathtt{mBHHO.E}(pk, m; \phi_i(r_1))$ to $\mathcal{A}$
LR query (m_0, m_1)
  $\mathcal{B}$ returns $\mathtt{mBHHO.E}(pk^*, m_b; r_1)$ to $\mathcal{A}$

When $\mathcal{A}$ halts and outputs a bit b', $\mathcal{B}$ halts and outputs 1 if and only if $b = b'$. If $g'_1, \ldots, g'_k = g_1^x, \ldots, g_k^x$, then $\mathcal{B}$ perfectly simulates G_0. If $g'_1, \ldots, g'_k = g^{u_1}, \ldots, g^{u_k}$, then $\mathcal{B}$ perfectly simulates G_1. Since the distributions of $g^{u_1}, \ldots, g^{u_k}$ and $g_1^{r_1}, \ldots, g_k^{r_k}$ are identical, we may conclude that

$$\begin{aligned}|\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_0^{\mathcal{A}} \Rightarrow 1]| &= |\mathbb{P}[\mathcal{B} \Rightarrow 1 | (g'_1, \ldots, \ldots, g'_k) = (g^{u_1}, \ldots, g^{u_k})] \\ &\quad - \mathbb{P}[\mathcal{B} \Rightarrow 1 | (g'_1, \ldots, \ldots, g'_k) = (g_1^x, \ldots, g_k^x)]| \\ &= \mathbf{Adv}^{k\text{-ddh}}_{\mathbb{G},\mathcal{B}}(\lambda).\end{aligned}$$

G_1 – G_2: We will show that if $\mathcal{A}$'s success probability is significantly different in games G_1 and G_2, then we can use $\mathcal{A}$ to build an adversary $\mathcal{C}$ that inverts the vector of functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_q)$. Specifically, we show:

Lemma 6. *For any polynomial-time adversary $\mathcal{A}$, we have:*

$$|\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1]| < \sqrt[3]{512\delta p^4 k}.$$

Here, recall that p is the size of the group $\mathbb{G}$ while $\boldsymbol{\phi}$ is a δ-hard-to-invert vector of functions.

Proof. If the success of $\mathcal{A}$ differs between games G_1 and G_2, then we can construct an adversary $\mathcal{C}$ that inverts the vector of functions $\boldsymbol{\phi}$ on input r_1. First, we consider an intermediate step. Suppose adversary $\mathcal{A}'$ is attempting to distinguish tuples of the form $T_1 = (g_1, \ldots, g_k, \prod_{i=1}^k g_i^{r_{1,i}}, \boldsymbol{\phi}(r_1), u, \langle r_1, u\rangle)$ from tuples of the form $T_2 = (g_1, \ldots, g_k, \prod_{i=1}^k g_i^{r_{1,i}}, \boldsymbol{\phi}(r_1), u, w)$, where w is uniformly random. If $\mathcal{A}$ can distinguish games 1 and 2 with probability ϵ, then $\mathcal{A}'$ can distinguish the previous two tuples with probability ϵ. The simulation runs as follows. Distinguisher $\mathcal{A}'$ is given a tuple $(g_1, \ldots, g_k, \prod_{i=1}^k g_i^{r_{1,i}}, \boldsymbol{\phi}(r_1), u, z)$, where z is either $\langle r_1, u\rangle$ or uniformly random. Then $\mathcal{A}'$ chooses a uniformly random generator g and bit b. $\mathcal{A}'$ uses the generators and the vector u to form a public key $pk^* = (g_1, \ldots, g_k, g^{u_1}, \ldots, g^{u_k})$, where u_i is the ith component of the vector u, and forwards this public key to $\mathcal{A}$. Then the distinguisher $\mathcal{A}'$ will answer the oracle queries of $\mathcal{A}$ as follows:

Enc query (pk, m, i)
  return $\mathtt{mBHHO.E}(pk, m; \phi(r_1))$ to $\mathcal{A}$
LR query (m_0, m_1)
  $c_1 \leftarrow \prod_{i=1}^k g_i^{r_{1,i}}$
  $(K, r) \leftarrow f(g^z)$
  $r' \leftarrow F_r(pk^* || m_b)$
  $c_2 \leftarrow \mathtt{DEM.E}(K, m_b; r')$
  return (c_1, c_2) to $\mathcal{A}$

When $\mathcal{A}$ outputs bit b', $\mathcal{A}'$ outputs 1 if and only if $b = b'$. It follows that

$$|\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1]| = |\mathbb{P}[\mathcal{A}'_{T_1} \Rightarrow 1] - \mathbb{P}[\mathcal{A}'_{T_2} \Rightarrow 1]|,$$

where $\mathcal{A}'_{T_i} \Rightarrow 1$ denotes that $\mathcal{A}'$ outputs 1 when given tuple T_i.

Now we shall use the distinguisher $\mathcal{A}'$ to construct our adversary $\mathcal{C}$ that will invert the vector of functions by using a modified version of Theorem 4.1 of [14]. The theorem shows how an adversary may invert a function when given access to a distinguisher that distinguishes tuples of the form $(\boldsymbol{\phi}(r_1), u, \langle r_1, u\rangle)$ from tuples of the form $(\boldsymbol{\phi}(r_1), u, w)$, where w is uniformly random. We therefore need a slight modification of their proof in order to invert the function when given tuples of the form T_1 or T_2 since these tuples require the extra parameters $g_1, \ldots, g_k, \prod_{i=1}^k g_i^{r_{1,i}}$. The inverter $\mathcal{C}$ can easily choose the generators, but the value $\prod_{i=1}^k g_i^{r_{1,i}}$ must be guessed by $\mathcal{C}$ (since he does not have the necessary information to form this value correctly), hence $\mathcal{C}$'s advantage is conditioned on the probability that he correctly supplies this value to $\mathcal{A}'$. The modified theorem we require is as follows:

Theorem 8. *Let p be a prime, let $\mathbb{G}$ be a group of size p, and let $H = \mathbb{Z}_2$. Let $\boldsymbol{\phi} : \mathbb{Z}_2^k \to \{0,1\}^*$ be a vector of functions, and let $g_1, \ldots, g_k$ be generators of $\mathbb{G}$. If there is a distinguisher $\mathcal{A}'$ that runs in time t such that*

$$|\mathbb{P}[r_1 \leftarrow \mathbb{Z}_2^k, u \leftarrow \mathbb{Z}_p^k, \{g_i\}_{i=1\ldots k} \leftarrow \mathbb{G}^k : \mathcal{A}'(g_1, \ldots, g_k, \prod_{i=1}^{k} g_i^{r_{1,i}}, \boldsymbol{\phi}(r_1), u, \langle r_1, u\rangle) \Rightarrow 1]$$

$$- \mathbb{P}[r_1 \leftarrow \mathbb{Z}_2^k, u \leftarrow \mathbb{Z}_p^k, \{g_i\}_{i=1\ldots k} \leftarrow \mathbb{G}^k, w \leftarrow \mathbb{Z}_p : \mathcal{A}'(g_1, \ldots, g_k, \prod_{i=1}^{k} g_i^{r_{1,i}}, \boldsymbol{\phi}(r_1), u, w) \Rightarrow 1]| = \epsilon$$

then there is an inverter $\mathcal{C}$ that runs in time $t' = \text{poly}(k, 2, 1/\epsilon)$ such that

$$\mathbb{P}[r_1 \leftarrow \mathbb{Z}_2^k : \mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1] \geq \frac{\epsilon^3}{512p^4k}.$$

Proof. The proof is very similar to that of Theorem 4.1 of [14], so we highlight only the modifications. Before $\mathcal{C}$ runs the simulation he first chooses generators $g_1, \ldots, g_k$ because these must be provided to $\mathcal{A}'$ (but they are not needed in [14]). Next, $\mathcal{C}$ needs to calculate the value $\prod_{i=1}^{k} g_i^{r_{1,i}}$ since this must also be given to $\mathcal{A}'$ (but is also not needed in [14]). Unfortunately, $\mathcal{C}$ does not have the required information to do this, so he randomly guesses an element $g^* \in \mathbb{G}$. Everything then proceeds as in the proof of Theorem 4.1 of [14], except when $\mathcal{C}$ is required to run $\mathcal{A}'$ he runs him with the extra inputs $g_1, \ldots, g_k, g^*$ (as well as the inputs that were required in [14]). If $\mathcal{C}$'s guess g^* for $\prod_{i=1}^{k} g_i^{r_{1,i}}$ is correct then $\mathcal{C}$ will provide tuples of the correct form to $\mathcal{A}'$. Hence, we condition on the probability that the guess is correct ($1/p$, the size of the group) and when the guess is correct we may use the same argument as Theorem 4.1 of [14] to bound $\mathcal{C}$'s advantage (notice that we ignore the probability of inverting when the guess is incorrect because the probability is always greater than or equal to zero and we only need a lower bound for $\mathcal{C}$).

$$\begin{aligned}
\mathbb{P}[\mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1] &= \frac{1}{p} \cdot \mathbb{P}[\mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1 \mid g^* = \prod_{i=1}^{k} g_i^{r_{1,i}}] + \frac{p-1}{p} \cdot \mathbb{P}[\mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1 \mid g^* \neq \prod_{i=1}^{k} g_i^{r_{1,i}}] \\
&> \frac{1}{p} \cdot \mathbb{P}[\mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1 \mid g^* = \prod_{i=1}^{k} g_i^{r_{1,i}}] \\
&\geq \frac{|\mathbb{P}[\mathcal{A}'_{T_1} \Rightarrow 1] - \mathbb{P}[\mathcal{A}'_{T_2} \Rightarrow 1]|^3}{512p^4k} \\
&= \frac{\epsilon^3}{512p^4k}.
\end{aligned}$$

Since $\mathcal{A}'$ has the same advantage as $\mathcal{A}$, we may conclude that

$$\mathbb{P}[\mathcal{C}(\boldsymbol{\phi}(r_1)) \Rightarrow r_1] > \frac{|\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1]|^3}{512p^4k}.$$

Furthermore, we must have

$$|\mathbb{P}[G_1^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1]| < \sqrt[3]{512\delta p^4 k}.$$

because otherwise $\mathcal{C}$ would invert the vector of functions with probability greater than δ, which is impossible by assumption.

$\mathbf{G_2}$ – $\mathbf{G_3}$: We will prove the following:

Lemma 7. *For any adversary $\mathcal{A}$, there exists a KDF adversary $\mathcal{D}$ such that:*

$$|\mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_3^{\mathcal{A}} \Rightarrow 1]| \leq \mathbf{Adv}^{\text{kdf}}_{f,\mathcal{D}}(\lambda).$$

Proof. The KDF adversary $\mathcal{D}$ against f with access to $\boldsymbol{\phi}$ will simulate either the game G_2 or G_3 for $\mathcal{A}$. Adversary $\mathcal{D}$ is given as input a value Z that is either a uniformly random value from the range of the KDF or an output of the KDF on a uniformly random value from the domain of the KDF. Adversary $\mathcal{D}$ simulates **proc. Initialise** in the $\boldsymbol{\phi}$-FV-RRA-CPA security game by setting $b \leftarrow_\$ \{0,1\}$, $r_1 \leftarrow_\$ \{0,1\}^k$ and by choosing and forwarding $pk^* = (g_1, \ldots, g_k, g^{u_1}, \ldots, g^{u_k})$ to $\mathcal{A}$. Adversary $\mathcal{D}$ then answers $\mathcal{A}$'s oracle queries as follows:

Enc query (pk, m, i)
  return $\texttt{mBHHO.E}(pk, m; \phi(r_1))$ to $\mathcal{A}$
LR query (m_0, m_1)
  $c_1 \leftarrow \prod_{i=1}^{k} g_i^{r_{1,i}}$
  $(K, r) \leftarrow Z$
  $r' \leftarrow F_r(pk^* || m_b)$
  $c_2 = \texttt{DEM.E}(K, m_b; r')$
  return (c_1, c_2) to $\mathcal{A}$

When $\mathcal{A}$ halts and outputs a bit b', $\mathcal{D}$ halts and outputs 1 if and only if $b = b'$. When $\mathcal{D}$ is given a random Z, it simulates G_3 perfectly. Otherwise, it provides a perfect simulation for G_2. Hence,

$$\begin{aligned}|\mathbb{P}[G_2^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_3^{\mathcal{A}} \Rightarrow 1]| &= |\mathbb{P}[\text{KDFReal}_f^{\mathcal{D}}(\lambda) \Rightarrow 1] - \mathbb{P}[\text{KDFRand}_{\$}^{\mathcal{D}}(\lambda) \Rightarrow 1]| \\ &= \mathbf{Adv}^{\text{kdf}}_{f,\mathcal{D}}(\lambda).\end{aligned}$$

G_3 – G_4: Notice that we now have a uniformly random output from the KDF in the **LR** queries, which means we now have a uniformly random key for the PRF in these queries. Hence, we can prove the following:

Lemma 8. *For any adversary $\mathcal{A}$, there exists a PRF adversary $\mathcal{E}$ such that:*

$$|\mathbb{P}[G_3^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_4^{\mathcal{A}} \Rightarrow 1]| \leq \mathbf{Adv}^{\text{prf}}_{F,\mathcal{E}}(\lambda).$$

Proof. The PRF adversary $\mathcal{E}$ will simulate either G_3 or G_4 for $\mathcal{A}$. Adversary $\mathcal{E}$ simulates **proc. Initialise** in the $\boldsymbol{\phi}$-FV-RRA-CPA security game by setting $b \leftarrow_{\$} \{0,1\}$, $r_1 \leftarrow_{\$} \{0,1\}^k$, $K \leftarrow_{\$} \texttt{DEM.K}(\lambda)$ and by choosing and forwarding $pk^* = (g_1, \ldots, g_k, g^{u_1}, \ldots, g^{u_k})$ to $\mathcal{A}$. Adversary $\mathcal{E}$ then answers $\mathcal{A}$'s oracle queries as follows:

Enc query (pk, m, i)
return $\texttt{mBHHO.E}(pk, m; \phi_i(r_1))$ to $\mathcal{A}$
LR query (m_0, m_1)
$c_1 = \prod_{i=1}^{k} g_i^{r_{1,i}}$
forward $pk^* || m_b$ to $\mathcal{E}$'s PRF oracle, receiving output r^*
$c_2 = \texttt{DEM.E}(K, m_b; r^*)$
return (c_1, c_2) to $\mathcal{A}$

When $\mathcal{A}$ halts and outputs b', $\mathcal{E}$ halts and outputs 1 if and only if $b = b'$. When $\mathcal{E}$ is playing the game PRFReal (in which case his oracle outputs are those of the PRF), he simulates G_3 perfectly. Otherwise, when $\mathcal{E}$ is playing the game PRFRand (and his outputs are uniformly random), he simulates G_4 perfectly. Hence,

$$\begin{aligned}|\mathbb{P}[G_3^{\mathcal{A}} \Rightarrow 1] - \mathbb{P}[G_4^{\mathcal{A}} \Rightarrow 1]| &= |\mathbb{P}[\text{PRFReal}_F^{\mathcal{E}}(\lambda) \Rightarrow 1] - \mathbb{P}[\text{PRFRand}_{\$}^{\mathcal{E}}(\lambda) \Rightarrow 1]| \\ &= \mathbf{Adv}^{\text{prf}}_{F,\mathcal{E}}(\lambda).\end{aligned}$$

Finally, since in handling $\mathcal{A}$'s **LR** queries, the outputs of the KDF f and the PRF F have now both been replaced with uniformly random values, we have uniformly random K and r''. Hence, the game G_4 may be simulated by a standard IND-CPA DEM adversary, $\mathcal{F}$. More formally:

Lemma 9. *For any adversary $\mathcal{A}$, there exists an IND-CPA DEM adversary $\mathcal{F}$ such that:*

$$2 \cdot \mathbb{P}[G_4^{\mathcal{A}} \Rightarrow 1] - 1 \leq \mathbf{Adv}^{\text{ind-cpa}}_{\texttt{DEM},\mathcal{F}}(\lambda).$$

Proof. The adversary $\mathcal{F}$ will simulate G_4 for $\mathcal{A}$. Adversary $\mathcal{F}$ chooses $r_1 \leftarrow_{\$} \{0,1\}^k$ and generates a public key pk^* of the form $(g_1, \ldots, g_k, g^{u_1}, \ldots, g^{u_k})$. Adversary $\mathcal{F}$ gives pk^* to $\mathcal{A}$, and answers $\mathcal{A}$'s oracles queries as follows:

Enc query (pk, m, i)
return $\texttt{mBHHO.E}(pk, m; \phi_i(r_1))$
LR query (m_0, m_1)
$c_1 = \prod_{i=1}^{k} g_i^{r_{1,i}}$
forward (m_0, m_1) to $\mathcal{F}$'s encryption oracle, receiving as output c_2
return (c_1, c_2) to $\mathcal{A}$

When $\mathcal{A}$ halts and outputs b', $\mathcal{F}$ halts and outputs b'. We conclude that

$$2 \cdot \mathbb{P}[G_4^{\mathcal{A}} \Rightarrow 1] - 1 \leq \mathbf{Adv}^{\text{ind-cpa}}_{\texttt{DEM},\mathcal{F}}(\lambda).$$

The theorem follows by combining all these inequalities.

The class of related randomness functions which our scheme `mBHHO` can tolerate is quite different from those in our previous constructions: linear and bounded-degree polynomials are certainly not hard-to-invert in general. Our proof of Theorem 7 actually shows that even if $\boldsymbol{\phi}(r)$ were to completely leak to the adversary (instead of merely being indirectly accessible via **Enc** queries), the scheme `mBHHO` would still be secure. This would not be the case if the analogous $\phi(r)$ values were to leak in our earlier schemes `PRF-PKE` and `CI-Hash-PKE`, since the adversary could actually reconstruct r from this leakage for the relevant ϕ functions and win the security game. Furthermore, the functions are not required to be collision-resistant or output-unpredictable. These restrictions are only strictly required of the functions queried to the **LR** oracle. However, since an an adversary is restricted to using only the identity function (which *is* collision-resistant and output-unpredictable) in its **LR** queries, the functions in $\boldsymbol{\Phi}$ do not need to satisfy these conditions.

References

1. Andrew Becherer, Alex Stamos, and Nathan Wilcox. Cloud computing security: Raining on the trendy new parade. *BlackHat USA*, 2009.
2. 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 Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 2009.
3. Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, 2010.
4. Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2011.
5. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.
6. Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 331–348. Springer, 2012.
7. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.
8. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay [33], pages 409–426.
9. Mike Bendel. Hackers describe PS3 security as epic fail, gain unrestricted access, 2011. `http://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/`.
10. Daniel J. Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko van Someren. Factoring RSA keys from certified smart cards: Coppersmith in the wild. Cryptology ePrint Archive, Report 2013/599, 2013. `http://eprint.iacr.org/`.
11. Bitcoin.org. Android security vulnerability, 2013. `http://bitcoin.org/en/alert/2013-08-11-android`.
12. Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2008.
13. Debian. Debian Security Advisory DSA-1571-1: OpenSSL – predictable random number generator, 2008. `http://www.debian.org/security/2008/dsa-1571`.
14. Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2010.
15. Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS*, pages 196–205. IEEE Computer Society, 2004.
16. Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. *IACR Cryptology ePrint Archive*, 2013:338, 2013.
17. Leo Dorrendorf, Zvi Gutterman, and Benny Pinkas. Cryptanalysis of the random number generator of the Windows operating system. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.
18. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
19. Ian Goldberg and David Wagner. Randomness and the Netscape browser, 1996. `http://www.drdobbs.com/windows/184409807`.
20. Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 182–200. Springer, 2011.
21. Zvi Gutterman and Dahlia Malkhi. Hold your sessions: An attack on java session-id generation. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2005.
22. Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Analysis of the linux random number generator. In *IEEE Symposium on Security and Privacy*, pages 371–385. IEEE Computer Society, 2006.
23. Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium*, August 2012.
24. Seny Kamara and Jonathan Katz. How to encrypt with a malicious random number generator. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 303–315. Springer, 2008.
25. Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 626–642. Springer, 2012.
26. Stefan Lucks. Ciphers secure against related-key attacks. In Roy and Meier [32], pages 359–370.
27. Kai Michaelis, Christopher Meyer, and Jörg Schwenk. Randomly failed! the state of randomness in current java implementations. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2013.
28. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

29. Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS*. The Internet Society, 2010.
30. Phillip Rogaway. Nonce-based symmetric encryption. In Roy and Meier [32], pages 348–359.
31. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Vaudenay [33], pages 373–390.
32. Bimal K. Roy and Willi Meier, editors. *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*. Springer, 2004.
33. Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.
34. Hoeteck Wee. Public key encryption against related key attacks. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2012.
35. Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2010.