Generalised Asynchronous Remote Key Generation for Pairing-based Cryptosystems

Nick Frymann[1], Daniel Gardham[1], Mark Manulis[2], and Hugo Nartz[2]

[1] Surrey Centre for Cyber Security, University of Surrey, Guildford, United Kingdom
n.frymann@surrey.ac.uk, daniel.gardham@surrey.ac.uk
[2] Research Institute CODE, Universität der Bundeswehr München, Munich, Germany mark@manulis.eu, hugo.nartz@unibw.de

Abstract. Asynchronous Remote Key Generation (ARKG, introduced in ACM CCS 2020) allows for a party to create public keys for which corresponding private keys may be later computed by another intended party only. ARKG can be composed with standard public-key cryptosystems and has been used to construct a new class of privacy-preserving proxy signatures. The original construction of ARKG, however, generates discrete logarithm key pairs of the form (x, g^x).

In this paper we define a generic approach for building ARKG schemes which can be applied to a wide range of pairing-based cryptosystems. This construction is based on a new building block which we introduce and call Asymmetric Key Generation (AKG) along with its extension ϕ-AKG where ϕ is a suitable mapping for capturing different key structures and types of pairings. We show that appropriate choice of ϕ allows us to create a secure ARKG scheme compatible with any key pair that is secure under the Uber assumption (EUROCRYPT 2004).

To demonstrate the extensive range of our general approach, we construct ARKG schemes for a number of popular pairing-based primitives: Boneh-Lynn-Shacham (JoC 2004), Camenisch-Lysyanskaya (CRYPTO 2004), Pointcheval-Sanders (CT-RSA 2016), Waters (EUROCRYPT 2005) signatures and structure-preserving signatures on equivalence classes (ASIACRYPT 2014). For each scheme we give an implementation and provide benchmarks that show the feasibility of our techniques.

Keywords Asynchronous Remote Key Generation, Pairings

1 Introduction

Asynchronous Remote Key Generation (ARKG) introduced by Frymann et al. [1] is a primitive that allows one party, following some initialisation step, to remotely create one or more public keys for another (receiving) party who can recover the corresponding secret keys at a later stage. As long as the key pairs generated via ARKG are compatible with some secure public key cryptosystem they can be securely used by the receiving party in that cryptosystem as its own key pair, e.g., to sign messages using a digital signature scheme or to decrypt ciphertexts

with some public key encryption scheme. This is ensured by the composability result from [1] and the two security properties of ARKG: SK-security ensures that only the designated receiving party can recover the derived private key; PK-unlinkability guarantees that the derived public keys cannot be linked. However, the only[3] existing ARKG construction is limited to cryptographic schemes based on the discrete-logarithm (DL) problem in a single group, i.e. of the form $(\mathsf{sk}, \mathsf{pk}) = (x, g^x)$ for some group generator g.

The ARKG primitive is useful in the context of decentralised applications where parties generate their own key pairs and require certain privacy guarantees. In fact the original ARKG construction was designed for WebAuthn [3] where it was used to enable back-up and recovery of WebAuthn credentials. In a nutshell, WebAuthn is a decentralised protocol which requires each user to have an independent key pair for each web account to perform a signature-based challenge-response authentication upon login while ensuring unlinkability across their accounts. In WebAuthn private keys are managed through authenticators that can be easily lost, in which case the user would be locked out of their accounts. ARKG helps to mitigate against this problem by allowing the user to use its current authenticator to pre-register public keys on web accounts for which it knows a long-term public key of the back-up authenticator (obtained from the initialisation step). The back-up authenticator can later recover the corresponding private keys and use them for authentication.

Although the original ARKG primitive was proposed for the application in WebAuthn, we observe that the actual instance of their protocol for key pairs of the form (x, g^x) has been deployed earlier in the context of stealth addresses for cryptocurrencies [4,5]. Following an initialisation step, during which a sender receives a long-term public key from another receiving party, it can transfer cryptocurrency to an ephemeral address (represented by an ephemeral public key) that it has created from the recipient's long-term public key without interaction with the recipient. The creation of this ephemeral public keys corresponds to the generation of public keys via ARKG. SK-security of ARKG property ensures that only the intended recipient is able to compute the corresponding private key and thus spend the cryptocurrency, whereas anonymity of the transaction would be implied by the PK-unlinkability property of ARKG.

As another application, ARKG has been used to construct a new class of privacy-preserving proxy signatures with unlinkable warrants [6]. These schemes adopt the delegation-by-warrant approach, yet in contrast to earlier schemes, delegated warrants and signatures produced by proxies remain unlinkable to the identities of the proxies. ARKG is the critical building block that performs the delegation step by creating an new verification key for the proxy using its long-term key for which the proxy can later compute the signing key. In this construction, unlinkability of warrants relies on the PK unkability of ARKG whereas the unforgeability property reduces to SK-security. This scheme has found applica-

[3] We note new ARKG constructions for lattice-based cryptosystems introduced concurrently by Frymann, Gardham, and Manulis at IEEE EuroS&P 2022 [2].

tions in decentralised environments, for example, enabling delegation of signing rights in the context of WebAuthn.

As mentioned previously, the original ARKG construction is restricted to DL-based keys in single groups which prevents the use of ARKG in cryptosystems that are not compatible with this setting. A prominent example of such cryptosystems can be found in pairing-based cryptography. Pairing-based cryptosystems enjoy flexibility and functionality over traditional group based setting and are widely used in privacy-preserving applications. This additional flexibility comes from the more complex nature that pairings introduce, namely the types of pairings, the key structures, and even the hardness assumptions can all vary hugely. Therefore, there can not be a single ARKG instance when we talk about pairing-based cryptosystems and instead a more general approach is required.

Contributions. The first main contribution is that we generalise the original DL-based ARKG construction by introducing a new building block which we call Asymmetric Key Generation (AKG) and its extension ϕ-AKG, where ϕ is an abstract map. We give a general transformation for building ARKG schemes from ϕ-AKG. In this way we not only provide a better understanding of the original DL-based ARKG scheme but also pave the way for new ARKG constructions.

The second main contribution is that we build first pairing-based ARKG schemes. Focusing on some particular type of pairings or some concrete structure of keys would be limiting. Instead, we develop pairing-based ϕ-AKG schemes by utilising the Uber assumption. By doing so, we can use our transformation to obtain many concrete instances of pairing-based ARKG schemes that would be able to cater for distinct types of pairings and different key structures. We prove the generic transformation has both SK-security and PK-unlinkability based on several properties (uniform sampling of private and public keys, one-time blindness, key secrecy) that we define for the underlying ϕ-AKG schemes.

To demonstrate extensive range that our techniques capture, we identify several concrete instances of ϕ-AKG based on both type-1 and type-2/3 pairings that rely on the hardness of standard assumptions. We use these to construct ARKG schemes for a wide range of signatures: BLS [7], Waters [8], CL [9], Structure-preserving signatures on equivalence classes [10] and Pointcheval-Sanders [11] signatures. We choose signatures that vary in their choice of pairing type, the format of the keys and the hardness assumptions on which their security relies, besides being quite popular in various privacy-preserving applications.

Organisation. First, we recall the formal definition of ARKG in Section 2. In Section 3 we define Asymmetric Key Generation (AKG) and the extension ϕ-AKG with their corresponding security properties. We then present our generic transformation from ϕ-AKG to ARKG. We prove that our construction satisfies PK-unlinkability and SK-security based on a new assumption that we call PRF-O_ϕ. In Section 4, we show that our PRF-O_ϕ assumption is implied by the Decisional Uber Assumption [12]. We construct various AKG schemes, from both type-1 and type-2/3 pairings and for variety of key structures. Finally in Section 5, we instantiate our generic transformation with suitable ϕ-AKG instances

to enable pairing-based ARKG schemes for several popular pairing-based signatures. We provide a publicly-available implementations and benchmark their performance.

2 Asynchronous Remote Key Generation

In this section we recall the syntax, model and security properties for ARKG. The original DL-based instantiation can be found in Figure 2.

2.1 The ARKG Model

Definition 1 (ARKG [1]). *An ARKG scheme is composed of five algorithms* $\mathtt{ARKG} := (\mathtt{Setup}, \mathtt{KGen}, \mathtt{DerivePK}, \mathtt{DeriveSK}, \mathtt{Check})$ *defined as follows:*

- $\mathtt{Setup}(1^\lambda)$: *This algorithm takes as input a security parameter* 1^λ*. It outputs a description* pp *of the public parameters for a security parameter* 1^λ*.*
- $\mathtt{KGen}(\mathsf{pp})$: *This algorithm takes as input public parameters* pp*. It outputs a private-public keypair* $(\mathsf{sk}, \mathsf{pk})$*.*
- $\mathtt{DerivePK}(\mathsf{pp}, \mathsf{pk}, aux)$: *This algorithm takes as input public parameters* pp*, a public key* pk *and auxiliary information aux. It probabilistically returns a public key* pk*' together with a link* cred *between* pk *and* pk'*.*
- $\mathtt{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$: *This algorithm takes as input public parameters* pp*, a secret key* sk *and credential* cred*. It either outputs the secret key* sk' *corresponding to* pk' *or* $\perp$ *on error.*
- $\mathtt{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}')$: *This algorithm takes as input public parameters* pp*, a secret key* sk*' and a public key* pk*'. It returns* 1 *if the keypair* $(\mathsf{sk}', \mathsf{pk}')$ *is legitimate, otherwise* 0*.*

We also recall the definition of correctness for ARKG. It states that asynchronously derived key pairs are as valid as freshly generated ones.

Correctness. An ARKG scheme is correct if it satisfies the following condition: For all $\lambda \in \mathbb{N}$ and $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$, the probability

$$\Pr\left[\mathtt{ARKG.Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}') = 1\right] = 1$$

if $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KGen}(\mathsf{pp})$, $(\mathsf{pk}', \mathsf{cred}) \leftarrow \mathtt{DerivePK}(\mathsf{pp}, \mathsf{pk}, \cdot)$ and $\mathsf{sk}' \leftarrow \mathtt{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$.

2.2 Security Definitions

An adversary $\mathcal{A}$ is modelled as a probabilistic polynomial time (PPT) algorithm allowed to call any of the above procedures. The security definitions introduced further also allow the adversary to interact with the primitive via oracles defined below.

– $\mathsf{O}_{\mathsf{pk'}}(\mathsf{pk}, \cdot)$: This oracle is parametrized with a public key pk and takes as input *aux*. It outputs the result of $\mathtt{DerivePK}(\mathsf{pp}, \mathsf{pk}, aux)$. These results are stored as $(\mathsf{pk}', \mathsf{cred})$ in a list PKList initialy set as $\varnothing$.
– $\mathsf{O}^{b}_{\mathsf{pk'}}(b, \mathsf{sk}_0, \mathsf{pk}_0)$: This oracle is parameterized with keypair $(\mathsf{sk}_0, \mathsf{pk}_0)$ and integer b, it takes no input. It outputs $(\mathsf{sk}', \mathsf{pk}')$ derived using $(\mathsf{sk}_0, \mathsf{pk}_0)$ when $b = 0$ or a uniformly sampled private-public key pair when $b = 1$.
– $\mathsf{O}_{\mathsf{sk'}}(\mathsf{sk}, \cdot)$: This oracle is parametrized with a secret key sk and takes a credential cred as input. It outputs the results of $\mathtt{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$ if $(\cdot, \mathsf{cred}) \in \mathsf{PKList}$, otherwise $\perp$. The results are stored as cred in a list SKList initialy set as $\varnothing$.

PK-unlinkability. This privacy property concerns the obfuscation of the link between pk and pk'. It ensures that derived key pairs $(\mathsf{sk}', \mathsf{pk}')$ are indistinguishable from fresh key pairs under the knowledge of pk. Formally, an ARKG scheme provides PK-unlinkability if the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}} = 1\right] - \frac{1}{2} \right|$$

where the PKU experiment is defined in Figure 1.

SK-security. This security property prevents unauthorized derivation of key pairs and credentials. We recall the four flavors introduced in [1] (mwKS, hwKS, msKS and hsKS) corresponding to malicious/honest and weak/strong variants of the security experiment $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{ARKG},\mathcal{A}}$. Formally, an ARKG scheme provides SK-security if the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{ARKG},\mathcal{A}} = 1\right]$$

where the KS experiment is defined in Figure 1.
Figure 2 contains the five algorithms introduced in [1] instantiating an ARKG scheme for DL-based keys. Public parameters pp contain a group $\mathbb{G}$ of order q with generator g, a Message Authentification Code MAC and two Key Derivation Functions KDF_1 and KDF_2 as defined in Appendix B.

3 Generalised ARKG

In this section we introduce and formally define Asymmetric Key Generation (AKG) schemes and their generalisation ϕ-AKG.

3.1 ϕ-AKG Schemes

Consider two asymmetric Diffie-Hellman (DH) key pairs $(s, S = g^s)$, $(e, E = g^e)$ as generated by the $\mathtt{ARKG.KGen}$ algorithm. These are used in the $\mathtt{ARKG.DerivePK}$ and $\mathtt{ARKG.DeriveSK}$ algorithms of Definition 1 to derive a value $\phi(S, e) = S^e = E^s = \phi(E, s)$ only available through knowledge of *crossed* key pair (e, S) or (s, E). We capture this notion in the next definition.

$\mathsf{Exp}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{Setup}$
2 : $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathtt{KGen}(\mathsf{pp})$
3 : $b \leftarrow\!\!\$\ \{0,1\}$
4 : $b' \leftarrow \mathcal{A}^{\mathsf{O}^{\mathsf{b}}_{\mathsf{pk}'}}(\mathsf{pp}, \mathsf{pk}_0)$
5 : **return** $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{ARKG},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{Setup}$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KGen}(\mathsf{pp})$
3 : $(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{pk}'}, \boxed{\mathsf{O}_{\mathsf{sk}'}}}(\mathsf{pp}, \mathsf{pk})$
4 : $\mathsf{sk}' \leftarrow \mathtt{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^\star)$
5 : **return** $\mathtt{Check}(\mathsf{pp}, \mathsf{sk}^\star, \mathsf{pk}^\star) \stackrel{?}{=} 1$
6 : $\quad \wedge\, \mathtt{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}^\star) \stackrel{?}{=} 1$
7 : $\quad \boxed{\wedge \mathsf{cred}^\star \notin \mathsf{SKList}}$
8 : $\quad \boxed{\wedge (\mathsf{pk}^\star, \mathsf{cred}^\star) \in \mathsf{PKList}}$

Fig. 1: The security experiments relating to PK-unlinkability on the left and SK-security on the right. $\boxed{\text{dashed}}$ boxes give strong variants (msKS, hsKS) of the KS security experiment while $\boxed{\text{dotted}}$ boxes give honest variants (hwKS, hsKS).

$\mathtt{Setup}$

1 : **return** $\mathsf{pp} = ((\mathbb{G}, g, q),$
2 : $\mathsf{MAC}, \mathsf{KDF}_1, \mathsf{KDF}_2)$

$\mathtt{KGen}(\mathsf{pp})$

1 : $x \leftarrow\!\!\$\ \mathbb{Z}_q$
2 : **return** $(\mathsf{pk}, \mathsf{sk}) = (x, g^x)$

$\mathtt{Check}(\mathsf{pp}, \mathsf{sk} = x, \mathsf{pk} = X)$

1 : **return** $g^x \stackrel{?}{=} X$

$\mathtt{DerivePK}(\mathsf{pp}, \mathsf{pk} = S, aux)$

1 : $(E, e) \leftarrow \mathtt{KGen}(\mathsf{pp})$
2 : $ck \leftarrow \mathsf{KDF}_1(S^e)$
3 : $mk \leftarrow \mathsf{KDF}_2(S^e)$
4 : $P \leftarrow g^{ck} \cdot S$
5 : $\mu \leftarrow \mathsf{MAC}(mk, (E, aux))$
6 : **return** $\mathsf{pk}' = P, \mathsf{cred} = (E, aux, \mu)$

$\mathtt{DeriveSK}(\mathsf{pp}, \mathsf{sk} = s, \mathsf{cred} = (E, \mu, aux))$

1 : $ck \leftarrow \mathsf{KDF}_1(E^s))$
2 : $mk \leftarrow \mathsf{KDF}_2(E^s)$
3 : **if** $\mu \stackrel{?}{=} \mathsf{MAC}(mk, (E, aux))$ **then**
4 : $\quad$ **return** $\mathsf{sk}' = \mathsf{sk} + s$
5 : **else return** $\perp$

Fig. 2: The original DL-based ARKG instantiation as defined in [1].

Definition 2 (ϕ-AKG). *An Asymmetric Key Generation (AKG) scheme is a tuple of algorithms* $\mathtt{AKG} := (\mathtt{Setup}, \mathtt{SKGen}, \mathtt{PKGen}, \mathtt{Check})$ *defined as:*

- $\mathtt{Setup}(1^\lambda)$: *This algorithm takes as input a security parameter* 1^λ*. It outputs a description* pp *of the public parameters of the scheme for security parameter* 1^λ*. The public parameters describe two groups* $\mathbb{G}_{\mathsf{sk}}$ *and* $\mathbb{G}_{\mathsf{pk}}$ *representing respectively the private and public key spaces.*
- $\mathtt{SKGen}(\mathsf{pp})$: *This algorithm takes as input public parameters* pp*. It computes and outputs a secret key* $\mathsf{sk} \in \mathbb{G}_{\mathsf{sk}}$*.*
- $\mathtt{PKGen}(\mathsf{pp}, \mathsf{sk})$: *This algorithm takes as input public parameters* pp *and a secret key* sk*. It computes and outputs a public key* $\mathsf{pk} \in \mathbb{G}_{\mathsf{pk}}$*.*
- $\mathtt{Check}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$: *This algorithm takes as input public parameters* pp*, a public key* pk *and a private key* sk*. It returns 1 if* $(\mathsf{pk}, \mathsf{sk})$ *forms a valid keypair, otherwise 0.*

Let ϕ *be an efficiently computable map* $\mathbb{G}_{\mathsf{sk}} \times \mathbb{G}_{\mathsf{pk}} \to \mathbb{G}$ *where* $\mathbb{G}$ *is an arbitrary group. An AKG in combination with* ϕ *and the following two algorithms* $\mathtt{SKCombine}$ *and* $\mathtt{SKInv}$ *form a* ϕ*-AKG scheme:*

- $\mathtt{SKCombine}(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2)$: *This algorithm takes as input public parameters* pp*, and two secret keys* sk_1 *and* sk_2*. It returns a secret key* $\mathsf{sk}' \in \mathbb{G}_{\mathsf{sk}}$*.*
- $\mathtt{SKInv}(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2)$: *This algorithm takes as input public parameters* pp*, and two secret keys* sk_1 *and* sk_2*. It returns a secret key* $\mathsf{sk}' \in \mathbb{G}_{\mathsf{sk}}$*.*

Remark 1. AKG schemes are implicit in many cryptosystems and usually bundle $\mathtt{SKGen}$ and $\mathtt{PKGen}$ together. However, the ARKG model requires *remote* key generation leading to this split. We write $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KGen}(\mathsf{pp})$ for $\mathsf{sk} \leftarrow \mathtt{SKGen}(\mathsf{pp}); \mathsf{pk} \leftarrow \mathtt{PKGen}(\mathsf{pp}, \mathsf{sk})$.

Correctness. An AKG scheme is correct if it satisfies the following condition: For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathtt{Setup}(1^\lambda)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KGen}(\mathsf{pp})$, we have

$$\Pr[\mathtt{AKG.Check}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}) = 1] = 1.$$

A ϕ-AKG scheme is correct if it is correct as an AKG scheme and if for every private-public key pairs $(\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2) \in \mathbb{G}_{\mathsf{pk}} \times \mathbb{G}_{\mathsf{sk}}$ the following three properties are satisfied:

$$\phi(\mathsf{sk}_1, \mathsf{pk}_2) = \phi(\mathsf{sk}_2, \mathsf{pk}_1) \tag{1}$$

$$\Pr[\mathtt{AKG.Check}(\mathsf{pp}, \mathtt{SKCombine}(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2), \mathsf{pk}_1 \cdot \mathsf{pk}_2) = 1] = 1. \tag{2}$$

$$\Pr[\mathsf{sk}_1 = \mathtt{SKInv}(\mathsf{pp}, \mathsf{sk}_2, \mathtt{SKCombine}(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2))] = 1. \tag{3}$$

Intuitively, property (1) states that crossed key pairs $(\mathsf{sk}_1, \mathsf{pk}_2)$ and $(\mathsf{sk}_2, \mathsf{pk}_1)$ can be used independently to derive a shared value. Property (2) states that the secret key corresponding to the product of public keys pk_1 and pk_2 in multiplicative group $\mathbb{G}_{\mathsf{pk}}$ can be efficiently computed from sk_1 and sk_2 using algorithm $\mathtt{SKCombine}$. For instance, let g be a generator of a group $\mathbb{G}$ of order p and suppose private-public key pairs are of type (x, g^x), then algorithm $\mathtt{SKCombine}$ simply performs addition over $\mathbb{Z}_p$. However for key pairs of type $(x, g^{x^2}) \in \mathbb{Z}_p \times \mathbb{G}$,

the algorithm should output a square root of $x^2 + y^2$ from inputs x and y in $\mathbb{Z}_p$. Property (3) allows for efficient inversion in the secret key group, this is required in the proofs of Theorems 2 and 3. In the rest of this paper, we assume AKG schemes to have the following 3 properties: uniform sampling for private and public keys (USK, UPK), one-time blindness (OTB) and key secrecy (KS) defined in the following. The corresponding experiments are defined in Figure 3.

Definition 3 (Uniform sampling). *We say that an AKG scheme with public and private key groups $\mathbb{G}_{\mathsf{pk}}, \mathbb{G}_{\mathsf{sk}}$ has uniform sampling if the following advantages are negligible in λ,*

$$\mathsf{Adv}^{\mathsf{UPK}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{UPK}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$
$$\mathsf{Adv}^{\mathsf{USK}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{USK}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

Definition 4 (Key Secrecy). *We say a ϕ-AKG scheme provides key secrecy (KS) if the following advantage is negligible in λ,*

$$\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}(\lambda) = 1\right].$$

Definition 5 (One-time blindness). *We say that a ϕ-AKG scheme with public and private key groups $\mathbb{G}_{\mathsf{pk}}, \mathbb{G}_{\mathsf{sk}}$ has the one-time blindness property if the following advantages are negligible in λ,*

$$\mathsf{Adv}^{\mathsf{OTB}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{OTB}}_{\mathbb{G},\mathcal{A}}(\lambda) = 1\right] \text{ for } \mathbb{G} \in \{\mathbb{G}_{\mathsf{pk}}, \mathbb{G}_{\mathsf{sk}}\}.$$

Example 1 (DL-based ϕ-AKG scheme). Let us outline the ϕ-AKG scheme used in the original ARKG implementation found in Figure 2. Let $\mathbb{G}$ be a cyclic group of order p generated by element g. Private keys are generated by sampling uniformally at random an element of $\mathbb{Z}_p$. A public key is generated from private key x by exponentiation g^x. This yields algorithms `SKGen` and `PKGen` for the common DL-based AKG scheme. Algorithms `Setup` and `Check` are also easily identified. Define further a mapping $\phi : \mathbb{G} \times \mathbb{Z}_p \to \mathbb{G}$ sending (g^x, y) to $(g^x)^y$. This mapping extends the AKG into a ϕ-AKG scheme. Algorithms `SKCombine` and `SKInv` are addition and inversion in group $(\mathbb{Z}_p, +)$ respectively. The uniform sampling of keys and OTB properties are verified by definition of the key generation process. Key secrecy follows for instance from the DL assumption in $\mathbb{G}$.

3.2 Our general transformation from ϕ-AKG to ARKG

Using the previously defined structures, we introduce a compiler transforming a ϕ-AKG scheme into an ARKG scheme. We first introduce some cryptographic primitives used in our construction. The definitions of Pseudorandom Function (PRF), Key Derivation Function (KDF) and Message Authentication Code (MAC) are given in Appendix B.

$\mathsf{Exp}^{\mathsf{USK}}_{\mathsf{AKG},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{AKG.Setup}(1^\lambda)$
2 : $\mathsf{sk}_0 \leftarrow \mathtt{AKG.SKGen}(\mathsf{pp})$
3 : $\mathsf{sk}_1 \leftarrow\!\!\$\ \mathbb{G}_{\mathsf{sk}}$
4 : $b \leftarrow\!\!\$\ \{0,1\}$
5 : $b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{sk}_b)$
6 : **return** $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{UPK}}_{\mathsf{AKG},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{AKG.Setup}(1^\lambda)$
2 : $\mathsf{sk} \leftarrow \mathtt{AKG.SKGen}(\mathsf{pp})$
3 : $\mathsf{pk}_0 \leftarrow \mathtt{AKG.PKGen}(\mathsf{pp}, \mathsf{sk})$
4 : $\mathsf{pk}_1 \leftarrow\!\!\$\ \mathbb{G}_{\mathsf{pk}}$
5 : $b \leftarrow\!\!\$\ \{0,1\}$
6 : $b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_b)$
7 : **return** $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{OTB}}_{\mathbb{G},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{AKG.Setup}$
2 : $x, y, z \leftarrow\!\!\$\ \mathbb{G}$
3 : **if** $\mathbb{G} == \mathbb{G}_{\mathsf{pk}}$
4 : $\quad (z_0, z_1) \leftarrow (x \cdot z, y \cdot z)$
5 : **else if** $\mathbb{G} == \mathbb{G}_{\mathsf{sk}}$
6 : $\quad z_0 \leftarrow \mathtt{SKCombine}(\mathsf{pp}, x, z)$
7 : $\quad z_1 \leftarrow \mathtt{SKCombine}(\mathsf{pp}, y, z)$
8 : $b \leftarrow\!\!\$\ \{0,1\}$
9 : $b \leftarrow \mathcal{A}(\mathsf{pp}, z_b, x, y)$
10 : **return** $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathtt{AKG.Setup}$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{AKG.KGen}(\mathsf{pp})$
3 : $\mathsf{sk}' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk})$
4 : **return** $\mathsf{sk}' \stackrel{?}{=} \mathsf{sk}$

Fig. 3: Uniform sampling of private and public keys, one-time blindness and key-secrecy experiments for ϕ-AKG schemes.

The lrPRF-O_ϕ assumption. In the DL setting, the PK-unlinkability property follows from the PRF-Oracle-Diffie-Hellman (PRF-ODH) assumption. This assumption and various flavours of it were introduced by Brendel et al. [13] to study TLS security. It is used to model a man-in-the-middle attack scenario where two parties derive a session key from an exchanged DH secret using a pseudorandom function PRF. Informally, the PRF-ODH assumption states that $\mathsf{PRF}(g^{uv}, \cdot)$ looks random even when knowing g^u, g^v and having access to values $\mathsf{PRF}(S^u, \cdot)$ and/or $\mathsf{PRF}(S^v, \cdot)$.

Definition 6 (Security under the lrPRF-O_ϕ assumption). *Let* $\mathrm{l}, \mathrm{r} \in \{\mathrm{n} = \mathit{none}, \mathrm{s} = \mathit{single}, \mathrm{m} = \mathit{many}\}$. *Let* $\mathsf{PRF} : \mathbb{G} \times L \to \mathbb{G}_{\mathsf{sk}}$ *be a pseudorandom function. We say* $(\phi\text{-}\mathtt{AKG}, \mathsf{PRF})$ *is* lrPRF-O_ϕ *secure if the following advantage is negligible in* λ *for all PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{lrPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{A}}(\lambda) = \left| \Pr\left[\mathit{Exp}^{\mathsf{lrPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{A}}(\lambda) = 1 \right] - \frac{1}{2} \right|,$$

where the lrPRF-O_ϕ *experiment is defined in Figure 4.*

$\mathsf{Exp}^{\mathsf{lrPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{A}}$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KGen}(\mathsf{pp})$
3 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KGen}(\mathsf{pp})$
4 : $\mathsf{bp} \leftarrow (\mathsf{bp}, \mathsf{pk}_1, \mathsf{pk}_2)$
5 : Pick challenge label $x \in \{0,1\}^*$
6 : $y_0 \leftarrow \mathsf{PRF}(\phi(\mathsf{pk}_2, \mathsf{sk}_1), x)$
7 : $y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
8 : $b \leftarrow\!\!\$\ \{0,1\}$
9 : $b' \leftarrow \mathcal{A}^{\mathsf{O}_\phi}(\mathsf{bp}, y_b, x)$
10 : **return** $b \stackrel{?}{=} b'$

$\mathsf{O}^l_\phi(S, \hat{x})$

1 : **if** $(S, \hat{x}) == (\mathsf{pk}_2, x)$ **then return** $\perp$
2 : **return** $\mathsf{PRF}(\phi(S, \mathsf{sk}_1), \hat{x})$

$\mathsf{O}^r_\phi(T, \hat{x})$

1 : **if** $(T, \hat{x}) == (\mathsf{pk}_1, x)$ **then return** $\perp$
2 : **return** $\mathsf{PRF}(\phi(T, \mathsf{sk}_2), \hat{x})$

Fig. 4: The $\mathsf{lrPRF\text{-}O}_\phi$ security experiment and its $\mathsf{O}^l_\phi, \mathsf{O}^r_\phi$ oracles. These oracles generalize the Diffie-Hellman oracles ODH_u and ODH_v in the lrPRF-ODH assumption [13].

Definition 7 (ϕ-AKG to ARKG compiler). *Our transformation of a ϕ-AKG scheme into an ARKG scheme uses two key derivation functions KDF_1 and KDF_2 and a message authentication code MAC. Function KDF_1 (resp. KDF_2) has input group $\mathbb{G}$ and target group $\mathbb{G}_{\mathsf{sk}}$ (resp. the input space of the MAC function). The algorithms of the resulting ARKG scheme are specified in Figure 5.*

We now proceed with the proof of PK-unklinkability and SK-security properties for the resulting scheme in Figure 5.

Theorem 1 (PK-unlinkability). *Let ϕ-$\mathtt{AKG}$ be a scheme providing key secrecy and let KDF_1, KDF_2 and MAC be functions with input and output spaces compatible with Figure 5. We assume these three functions are secure under the definitions given in Appendix B. If ϕ-$\mathtt{AKG}$ is secure under the $\mathsf{nnPRF\text{-}O}_\phi$ assumption, the compiled ARKG scheme satisfies PK-unlinkability.*

Proof. Let game $\mathcal{G}_0$ be defined by the $\mathsf{Exp}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}$ experiment. Thus, $\Pr[\mathcal{G}_0 = 1] = \mathsf{Adv}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$. Recursively define a series of hybrid games by $\mathcal{H}_0 = \mathcal{G}_0$ and $\mathcal{H}_i = \mathcal{H}_{i-1}$ with the exception that on the i-th oracle call to $\mathsf{O}_{\mathsf{pk}'}$:

- computation of the public key pk' is replaced by $\mathtt{AKG.PKGen}(\mathsf{pp}, ck) \cdot R$,
- computation of the secret key sk' is replaced by $\mathtt{AKG.SKCombine}(\mathsf{pp}, ck, r)$

where $(R, r) \leftarrow \mathtt{AKG.KGen}(\mathsf{pp})$.

Assume $\mathcal{A}$ is able to distinguish between the two games $\mathcal{H}_i$ and $\mathcal{H}_{i-1}$. Let $\mathcal{B}$ be an adversary for experiment $\mathsf{Exp}^{\mathsf{OTB}}_{\mathbb{G}_{\mathsf{sk}},\mathcal{A}}$ receiving challenge (pp, z_b, x, y). Adversary $\mathcal{B}$ wins if it is able to tell whether $z_b = \mathtt{SKCombine}(\mathsf{pp}, x, z)$ or $z_b = \mathtt{SKCombine}(\mathsf{pp}, y, z)$ for some uniformly sampled element z. The USK and UPK assumptions in $\mathbb{G}_{\mathsf{sk}}$ and $\mathbb{G}_{\mathsf{pk}}$ make keys indistinguishable from uniform sampling. The following distributions are therefore indistinguishable

$\texttt{Setup}(1^\lambda)$

1 : $\tilde{\mathsf{pp}} \leftarrow \texttt{AKG.Setup}(1^\lambda)$
2 : **return** $\mathsf{pp} = \tilde{\mathsf{pp}}$

$\texttt{KGen}(\mathsf{pp})$

1 : **return** $(\mathsf{pk}, \mathsf{sk}) = \texttt{AKG.KGen}(\mathsf{pp})$

$\texttt{Check}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$

1 : **return** $\texttt{AKG.Check}(\mathsf{pk}, \mathsf{sk})$

$\texttt{DerivePK}(\mathsf{pp}, S, aux)$

1 : $(e, E) \leftarrow \texttt{AKG.KGen}(\mathsf{pp})$
2 : $ck \leftarrow \mathsf{KDF}_1(\phi(S, e))$
3 : $mk \leftarrow \mathsf{KDF}_2(\phi(S, e))$
4 : $P \leftarrow \texttt{AKG.PKGen}(\mathsf{pp}, ck) \cdot S$
5 : $\mu \leftarrow \texttt{MAC.Tag}(mk, (E, aux))$
6 : **return** $\mathsf{pk}' = P, \mathsf{cred} = (E, aux, \mu)$

$\texttt{DeriveSK}(\mathsf{pp}, s, \mathsf{cred} = (E, \mu, aux))$

1 : $ck \leftarrow \mathsf{KDF}_1(\phi(s, E))$
2 : $mk \leftarrow \mathsf{KDF}_2(\phi(s, E))$
3 : **if** $\texttt{MAC.Verify}(mk, (E, aux), \mu) \stackrel{?}{=} 1$ **then**
4 : **return** $\mathsf{sk}' = \texttt{AKG.SKCombine}(\mathsf{pp}, ck, s)$
5 : **else return** $\perp$

Fig. 5: Our general transformation from ϕ-AKG to ARKG.

- (x, y, z) and (s, r, ck)
- $(S, R, \texttt{AKG.PKGen}(\mathsf{pp}, ck))$ and
  $(\texttt{AKG.PKGen}(\mathsf{pp}, x), \texttt{AKG.PKGen}(\mathsf{pp}, y), \texttt{AKG.PKGen}(\mathsf{pp}, z))$.

As such, adversary $\mathcal{B}$ can ask $\mathcal{A}$ to distinguish between $\mathcal{H}'$ and $\mathcal{H}''$ where

- $\mathcal{H}' = \mathcal{H}_{i-1}$ except $s = x$, $ck = z$ and $S = \texttt{AKG.PKGen}(\mathsf{pp}, x)$ and
- $\mathcal{H}'' = \mathcal{H}_i$ except $r = y$, $ck = z$ and $S = \texttt{AKG.PKGen}(\mathsf{pp}, r)$.

The result is a distinguisher for the OTB experiment, which is supposed hard. Thus $\Pr[\mathcal{H}_i = 1] = \Pr[\mathcal{H}_{i-1} = 1]$. Set game $\mathcal{G}_1$ as $\mathcal{H}_q$ where q is the last oracle call index. Recursively define another series of games as $\tilde{\mathcal{H}}_0 = \mathcal{G}_1$ and $\tilde{\mathcal{H}}_i = \tilde{\mathcal{H}}_{i-1}$ with the exception that on the i-th oracle call to $\mathsf{O}_{\mathsf{pk}'}$ expressions '$\phi(S, e)$' and '$\phi(E, s)$' are replaced with 'u' where $u \leftarrow\!\!\$\ \mathbb{G}_{\mathsf{pk}}$. The nnPRF-$\mathsf{O}_\phi$ assumption coupled with properties of PRF function KDF_1 ensure that games $\tilde{\mathcal{H}}_i$ and $\tilde{\mathcal{H}}_{i-1}$ are indistinguishable. Thus $\Pr\left[\tilde{\mathcal{H}}_i = 1\right] = \Pr\left[\tilde{\mathcal{H}}_{i-1} = 1\right]$.
Now assume $\mathcal{A}$ is able to win at the PK-unlinkability game. We construct an adversary $\mathcal{B}$ for the nnPRF-O_ϕ game that wins with non-negligible probability. Adversary $\mathcal{B}$ plays the role of challenger for $\mathcal{A}$ in $\tilde{\mathcal{H}}_j$. It invokes its own nnPRF-O_ϕ game, receiving pk_1, pk_2, y_c and sets the label of PRF to the one of KDF_1. The game is won if $\mathcal{B}$ can correctly guess bit c.

Adversary $\mathcal{B}$ sets up game $\tilde{\mathcal{H}}_j$ for $\mathcal{A}$ with $\mathsf{pk}_0 \leftarrow \mathsf{pk}_1$ and $\mathsf{sk}_0 \leftarrow \perp$. It answers $\mathcal{A}$'s j-th oracle query to $\mathsf{O}^{\mathsf{b}}_{\mathsf{pk}'}$ honestly except it sets $\mathsf{pk}' = \mathsf{pk}_2$, $\mathsf{sk}' = \perp$ and $ck_j = y_j$, the output of KDF_1 in game $\tilde{\mathcal{H}}_j$. It then waits for $\mathcal{A}$ to produce a bit b and forwards it to nnPRF-O_ϕ. The distribution of sk_2 in the nnPRF-O_ϕ experiment is the same as the distribution of e in $\texttt{ARKG.DerivePK}$. Thus the distributions of

$\phi(\mathsf{pk}_1, \mathsf{sk}_2)$ and $\phi(\mathsf{pk}_1, e)$ are equal. As such, $\mathcal{B}$ wins with probability equal to that of $\mathcal{A}$ distinguishing between $\tilde{\mathcal{H}}^b_j$ and $\tilde{\mathcal{H}}^b_{j-1}$ and $\Pr\left[\tilde{\mathcal{H}}^b_i = 1\right] = \Pr\left[\tilde{\mathcal{H}}^b_{i-1} = 1\right]$. Set $\mathcal{G}_2 = \tilde{\mathcal{H}}^b_q$ where q is the last oracle call index. We define yet another series of games as $\hat{\mathcal{H}}_0 = \mathcal{G}_2$ and $\hat{\mathcal{H}}_i = \hat{\mathcal{H}}_{i-1}$ with the exception that on the i-th oracle call to $\mathsf{O}_{\mathsf{pk}'}$ line '$mk \leftarrow \mathsf{KDF}_2(\phi(S, e))$' is replaced with '$mk \leftarrow \mathsf{KDF}_2(u)$' where $u \leftarrow\!\!\$\ \mathbb{G}_T$. An argument similar to the one showing $\tilde{\mathcal{H}}^b_j$ is indistinguishable from $\tilde{\mathcal{H}}^b_{j-1}$ shows that $\hat{\mathcal{H}}^b_j$ is indistinguishable from $\tilde{\mathcal{H}}^b_{j-1}$, $\Pr\left[\hat{\mathcal{H}}^b_i = 1\right] = \Pr\left[\hat{\mathcal{H}}^b_{i-1} = 1\right]$. Set $\mathcal{G}_3 = \hat{\mathcal{H}}^b_q$ where q is the last oracle call index. The advantage in distinguishing between $b = 0$ and $b = 1$ is equal to $1/2$ as the output distributions pk' and sk' are now identical and independent from b, thus $\Pr[\mathcal{G}_3 = 1] = \frac{1}{2}$. Finally we get a bound for the advantage of $\mathcal{A}$ in the PKUs experiment

$$\mathsf{Adv}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \left(\mathsf{Adv}^{\mathsf{nnPRF\text{-}O}_\phi}_{\mathsf{KDF}_1,\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{nnPRF\text{-}O}_\phi}_{\mathsf{KDF}_2,\mathcal{A}}(\lambda) \right).$$

According to our assumptions, $\mathsf{Adv}^{\mathsf{PKU}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$ is negligible in λ. □

Theorem 2 (hsKS-security). *Let ϕ-AKG be a scheme providing key secrecy and let KDF_1, KDF_2 and MAC be functions with input and output spaces compatible with Figure 5. We assume these three functions are secure under the definitions given in Appendix B. The compiled ARKG scheme is* hsKS*-secure (and therefore* hwKS*-secure).*

Proof. Define $\mathcal{G}_0$ as $\mathsf{Exp}^{\mathsf{hsKS}}_{\mathcal{A}}$ and $\mathcal{G}_1$ as $\mathcal{G}_0$ except

- During oracle calls to $\mathsf{O}_{\mathsf{pk}'}$, line 4 of `DerivePK` is replaced with

$$r \leftarrow\!\!\$\ \mathbb{G}_{\mathsf{sk}}; \quad P \leftarrow \mathtt{AKG.PKGen}(\mathsf{pp}, c) \cdot \mathtt{AKG.PKGen}(\mathsf{pp}, r).$$

- If $\mathcal{A}$ queries $\mathsf{O}_{\mathsf{sk}'}$ with $E \in \mathsf{cred}$ such that $E \in \mathsf{List}$, then line 4 of `DeriveSK` is replaced with "**return** $\mathsf{sk}' = \mathtt{SKCombine}(\mathsf{pp}, ck, r)$". This ensures the validity of the derived keys.

The two games $\mathcal{G}_0$ and $\mathcal{G}_1$ are indistinguishable as s and r are uniformly sampled from the same space by the PKU assumption.

We now construct an adversary $\mathcal{B}$ for the $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}$ game assuming an adversary $\mathcal{A}$ is able to win game hsKS with non-negligible probability. Adversary $\mathcal{B}$ instantiates its own $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}$ experiment, receiving challenge $\mathsf{pk} = S \leftarrow \mathtt{AKG.PKGen}(\mathsf{pp}, s)$. It wins if it is able to recover the secret key $\mathsf{sk} = s$.

$\mathcal{B}$ sets up the hsKS game as described in the experiment but replaces line 2 with '$(\mathsf{pk}, \mathsf{sk}) \leftarrow (S, \perp)$'. Adversary $\mathcal{B}$ chooses an oracle query where it guesses $\mathcal{A}$ will use the derived key pk' in its forgery. For this single query, $\mathcal{B}$ can answer calls to $\mathsf{O}_{\mathsf{pk}'}$ using S but cannot answer calls to $\mathsf{O}_{\mathsf{sk}'}$. Should this later call be queried, the experiment aborts. In the other cases, $\mathcal{B}$ can answer calls to $\mathsf{O}_{\mathsf{sk}'}$ as it generates the ephemeral keys (e, E) and can locate r using the list kept by the oracle.

Adversary $\mathcal{B}$ waits for a successful forgery $(\mathsf{sk}', \mathsf{cred}^\star)$ from $\mathcal{A}$. Using $\mathsf{cred}^\star$, it can locate (e, E) corresponding to $\mathsf{cred}^\star$ in the list. As such, it is able to compute ck and $s = \mathtt{AKG.SKInv}(\mathsf{pp}, \mathsf{sk}', ck)$. Adversary $\mathcal{B}$ is guaranteed to find (E, e) in List as successful forgery of a tuple for hsKS requires, following line 8 of the experiment's definition, that $(\mathsf{pk}^\star, \mathsf{cred}^\star) \in \mathsf{PKList}$, which implies a call to $\mathsf{O}_{\mathsf{pk}'}$. However, the experiment fails if $\mathsf{O}_{\mathsf{sk}'}$ is called with pk', which happens with probability $\epsilon = 1/\#\mathbb{G}_{\mathsf{pk}}$.

Thus, the advantage of $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{hsKS}}_{\mathcal{A}}$ is bounded by the advantage of an adversary $\mathcal{B}$ against the $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}$ game by

$$\mathsf{Adv}^{\mathsf{hsKS}}_{\mathcal{A}}(\lambda) \leqslant (1-\varepsilon)\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}.$$

By assumption the $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{AKG},\mathcal{A}}$ experiment is hard and it follows that the ARKG scheme is hsKS-secure. □

Theorem 3 (msKS-security). *Let ϕ-$\mathtt{AKG}$ be a scheme providing key secrecy and let KDF_1, KDF_2 and MAC be functions with input and output spaces compatible with Figure 5. We assume these three functions are secure under the definitions given in Appendix B. If ϕ-$\mathtt{AKG}$ is secure under the snPRF-O_ϕ assumption, the compiled ARKG scheme is msKS-secure (and therefore mwKS-secure).*

Proof. See Appendix A.1.

4 Pairing-based ϕ-AKG and ARKG schemes from Uber assumption

In this section we define generic ϕ-AKG schemes for various key structures over bilinear groups so that they can be used with the transformation in Figure 5. Using the decisional Uber assumption [12, 14], we provide a generic result on the security of the ARKG instance obtained through this transformation in Lemma 1. We use this result to prove the security of ARKG instances obtained from ϕ-AKG schemes based on the three types of pairings.

4.1 Notations and Building Blocks

Arrows over letters will indicate vectors. Let us fix integers n, m, k, l and denote the polynomial ring $\mathbb{Z}_n[X_1, \cdots, X_m]$ by the letter A. For any vectors of m-variate polynomials $\boldsymbol{F} \in A^k$, $\boldsymbol{H} \in A^l$ and vector $\boldsymbol{x} \in \mathbb{Z}_n^m$, we write $\boldsymbol{F}(\boldsymbol{x}) := (F_1(\boldsymbol{x}), \cdots, F_k(\boldsymbol{x}))$. Similarily, for a group $\mathbb{G}$ of order n and element $g \in \mathbb{G}$ we write $g^{\boldsymbol{F}(\boldsymbol{x})} := (g^{F_1(\boldsymbol{x})}, \cdots, g^{F_k(\boldsymbol{x})})$. We concatenate vectors with nested notation:

$$(g^{\boldsymbol{F}(\boldsymbol{x})}, g^{\boldsymbol{H}(\boldsymbol{x})}) := (g^{F_1(\boldsymbol{x})}, \cdots, g^{F_k(\boldsymbol{x})}, g^{H_1(\boldsymbol{x})}, \cdots, g^{H_l(\boldsymbol{x})}) \in \mathbb{G}^{k+l}.$$

Finally for a binary vector $\boldsymbol{\epsilon} \in \{0,1\}^m$, we write $\boldsymbol{x}^{\boldsymbol{\epsilon}} = (x_1^{\epsilon_1}, \cdots, x_m^{\epsilon_m})$. In this section and the next, when using mappings $\phi(\mathsf{sk}_1, \mathsf{pk}_2)$, we denote by x

(possibly with sub-indices) exponents belonging to the first key pair $(\mathsf{sk}_1, \mathsf{pk}_1)$ and y for exponents of the second key pair $(\mathsf{sk}_2, \mathsf{pk}_2)$. For instance, we will write $\phi(\mathsf{sk}_1, \mathsf{pk}_2) = e(g_1^{x_1}, g_2^{y_1})$, which stands for $\phi(\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y}))$, without re-introducing vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

Definition 8 (Bilinear Groups). *A description of a bilinear group $\mathcal{G}$ is a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ such that*

- *$\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p,*
- *$\mathbb{G}_1$ (resp. $\mathbb{G}_2$) is generated by element g_1 (resp. g_2),*
- *$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear pairing,*
- *$\gamma : \mathbb{G}_2 \to \mathbb{G}_1$ is an isomorphism.*

In the above definition, *non-degenerate bilinear pairing* means group homomorphism linear in both components and such that neither $e(g_1, \cdot)$ nor $e(\cdot, g_2)$ are trivial maps. We assume group operations as well as mapping e to be efficiently computable. Assumptions on the efficient computability of γ and γ^{-1} give rise to three types of bilinear groups:

- Type 1: both γ and γ^{-1} are efficiently computable,
- Type 2: γ is efficiently computable but γ^{-1} is not,
- Type 3: Neither γ nor γ^{-1} is efficiently computable.

Below we recall the DBDH, XDH and SXDH assumptions for bilinear pairings.

Definition 9 (DBDH, XDH, SXDH). *Let $\mathcal{G}$ be a bilinear group. For $\mathsf{A} \in \{\mathit{DBDH}, \mathit{XDH}, \mathit{SXDH}\}$ we say assumption A holds in $\mathcal{G}$ if the following advantage is negligible in λ,*

$$\mathsf{Adv}^{\mathsf{A}}_{\mathcal{G},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{A}}_{\mathcal{G},\mathcal{A}}(\lambda) = 1\right]$$

where the corresponding experiments are defined in Figure 6.

We now give a specialization of Definition 2 to schemes using bilinear pairings. This particular description provides a framework for schemes with private-public key pairs consisting of order elements and bilinear group elements. It is most convenient when used in conjunction with the Uber assumption [12, 15] and encompasses a wide range of key types that are commonly used in pairing-based cryptosystems.

Definition 10 (Pairing-based ϕ-AKG schemes). *A pairing-based ϕ-AKG is based on a bilinear group $\mathcal{G}$ and an* `AKG` *scheme. It is defined as follows. Let $(m, l_F, l_H, l_K, k_F, k_H, k_K)$ be integers and $\boldsymbol{\epsilon} = (\epsilon_1, \cdots, \epsilon_m)$ be a binary vector. Let $(\boldsymbol{F}^{\mathsf{sk}}, \boldsymbol{H}^{\mathsf{sk}}, \boldsymbol{K}^{\mathsf{sk}}) \in A^{l_F + l_H + l_K}$ and $(\boldsymbol{F}^{\mathsf{pk}}, \boldsymbol{H}^{\mathsf{pk}}, \boldsymbol{K}^{\mathsf{pk}}) \in A^{k_F + k_H + k_K}$ be vectors of m-variate polynomials over $\mathbb{Z}_p$. Assume private-public key pairs of* `AKG` *are parametrized by exponents via*

$$\mathsf{sk}(\boldsymbol{x}) = \left(\boldsymbol{x}^{\boldsymbol{\epsilon}}, g_1^{\boldsymbol{F}^{\mathsf{sk}}(\boldsymbol{x})}, g_2^{\boldsymbol{H}^{\mathsf{sk}}(\boldsymbol{x})}, g_T^{\boldsymbol{K}^{\mathsf{sk}}(\boldsymbol{x})}\right),$$
$$\mathsf{pk}(\boldsymbol{x}) = \left(g_1^{\boldsymbol{F}^{\mathsf{pk}}(\boldsymbol{x})}, g_2^{\boldsymbol{H}^{\mathsf{pk}}(\boldsymbol{x})}, g_T^{\boldsymbol{K}^{\mathsf{pk}}(\boldsymbol{x})}\right)$$

$\mathsf{Exp}^{\mathsf{DBDH}}_{\mathcal{G},\mathcal{A}}$

$(x, y, z) \leftarrow\!\$\ \mathbb{Z}_p^3$
$c_0 \leftarrow e(g_1, g_2)^{xyz}$
$c_1 \leftarrow\!\$\ \mathbb{G}_T$
$b \leftarrow\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}(\mathcal{G}, (g_1^x, g_1^y, g_1^z, c_b))$
return $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{XDH}}_{\mathcal{G},\mathcal{A}}$

$(x, y) \leftarrow\!\$\ \mathbb{Z}_p$
$c_0 \leftarrow g_1^{xy}$
$c_1 \leftarrow\!\$\ \mathbb{G}_1$
$b \leftarrow\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}(\mathcal{G}, (g_1^x, g_1^y, c_b))$
return $b \stackrel{?}{=} b'$

$\mathsf{Exp}^{\mathsf{SXDH}}_{\mathcal{G},\mathcal{A}}$

$(x, y, z, t) \leftarrow\!\$\ \mathbb{Z}_p$
$(c_0, d_0) \leftarrow (g_1^{xy}, g_2^{zt})$
$(c_1, d_1) \leftarrow\!\$\ \mathbb{G}_1 \times \mathbb{G}_2$
$(a, b) \leftarrow\!\$\ \{0,1\}^2$
$(a', b') \leftarrow \mathcal{A}(\mathcal{G}, (g_1^x, g_1^y, c_a), (g_2^z, g_2^t, d_b))$
return $(a \stackrel{?}{=} a') \wedge (b \stackrel{?}{=} b')$

Fig. 6: The Decisional Bilinear Diffie-Hellman (DBDH), Extended Diffie-Hellman (XDH) and Symmetric Extended Diffie-Hellman (SXDH) experiments.

for vector $\boldsymbol{x} = (x_1, \cdots, x_m) \in \mathbb{Z}_p$.
Let $Q \in \mathbb{Z}_p[X_1 \cdots, X_m, Y_1, \cdots, Y_m]$ *be a* $2m$*-variate polynomial and define* $\phi : (\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y})) \mapsto g_T^{Q(\boldsymbol{x},\boldsymbol{y})}$. *We write* $(\epsilon, \boldsymbol{F}^{\mathsf{sk}}, \boldsymbol{H}^{\mathsf{sk}}, \boldsymbol{K}^{\mathsf{sk}} | \boldsymbol{F}^{\mathsf{pk}}, \boldsymbol{H}^{\mathsf{pk}}, \boldsymbol{K}^{\mathsf{pk}} | Q)$ *as a description of the scheme.*

Such a ϕ-AKG scheme has the correctness property 1 of Definition 2 if, for every pair of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^m$, the value $g_T^{Q(\boldsymbol{x},\boldsymbol{y})}$ is efficiently computable from the knowledge of either one of crossed key pairs $(\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y}))$ and $(\mathsf{sk}(\boldsymbol{y}), \mathsf{pk}(\boldsymbol{x}))$. The decisional Uber assumption provides a general framework encompassing many standard assumptions. We use it to prove the PK-unlinkability property of an ARKG instance obtained from a pairing-based ϕ-AKG scheme through the transformation in Figure 5. We recall the decisional Uber assumption and prove this result in Lemma 1.

Definition 11 (Decisional Uber Assumption [12, 15]). *Let* $\mathcal{G}$ *be a bilinear group description. Fix integers* f, h, k, m. *Take three* m*-variate polynomials vectors* $\boldsymbol{F} = (F_1, \cdots, F_f)$, $\boldsymbol{H} = (H_1, \cdots, H_h)$, $\boldsymbol{K} = (K_1, \cdots, K_k)$ *and a target polynomial* Q. *The decisional* $(\boldsymbol{F}, \boldsymbol{H}, \boldsymbol{K}, Q)$-Uber *assumption is said to hold in bilinear group* $\mathcal{G}$ *if, for the* $\mathsf{Exp}^{\mathsf{Uber}}_{\mathcal{A}}$ *experiment described in Figure 7, the following advantage is negligible in* λ*:*

$$\mathsf{Adv}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$

$\mathsf{Exp}^{\mathsf{Uber}}_{\mathcal{A}}$

$\boldsymbol{x} \leftarrow\!\!\$\ \mathbb{Z}_p^m$

$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \leftarrow g_1^{\boldsymbol{F}(\boldsymbol{x})}, g_2^{\boldsymbol{H}(\boldsymbol{x})}, g_T^{\boldsymbol{K}(\boldsymbol{x})}$

$y_0 = g_T^{Q(\boldsymbol{x})}$

$y_1 \leftarrow\!\!\$\ \mathbb{G}_T$

$b \leftarrow\!\!\$\ \{0,1\}$

$b' \leftarrow \mathcal{A}(\mathsf{pp}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, y_b)$

return $b \stackrel{?}{=} b'$

Fig. 7: The Uber experiment. Adversary $\mathcal{A}$ wins if it is able to distinguish target value $g_T^{Q(\boldsymbol{x})}$ from a uniformly sampled element in $\mathbb{G}_T$.

Let ϕ-AKG be a pairing-based scheme as defined in Definition 10 and let PRF be a pseudorandom function with input space $\mathbb{G}_T$. In the following we prove that the nnPRF-O_ϕ security of (ϕ-AKG, PRF) according to Definition 6 is implied by a certain parameterisation of the decisional Uber assumption. This parameterisation depends on $2m$-variate polynomials in $X_1, \cdots, X_m, Y_1, \cdots, Y_m$ for which we also write $X = (X_1, \cdots, X_m)$ and $Y = (Y_1, \cdots, Y_m)$. From ϕ-AKG define $2m$-variate polynomial vectors

$$\begin{aligned}
\boldsymbol{F} &= (\boldsymbol{F}_1^{\mathsf{pk}}(X), \cdots, \boldsymbol{F}_{l_F}^{\mathsf{pk}}(X), \boldsymbol{F}_1^{\mathsf{pk}}(Y), \cdots, \boldsymbol{F}_{l_F}^{\mathsf{pk}}(Y)) \\
\boldsymbol{H} &= (\boldsymbol{H}_1^{\mathsf{pk}}(X), \cdots, \boldsymbol{H}_{l_H}^{\mathsf{pk}}(X), \boldsymbol{H}_1^{\mathsf{pk}}(Y), \cdots, \boldsymbol{H}_{l_H}^{\mathsf{pk}}(Y)) \\
\boldsymbol{K} &= (\boldsymbol{K}_1^{\mathsf{pk}}(X), \cdots, \boldsymbol{K}_{l_K}^{\mathsf{pk}}(X), \boldsymbol{K}_1^{\mathsf{pk}}(Y), \cdots, \boldsymbol{K}_{l_K}^{\mathsf{pk}}(Y)).
\end{aligned}$$

The m-variate polynomials defining ϕ-AKG appear twice: once in the variables $X_1, \cdots, X_m$ and once in variables $Y_1, \cdots, Y_m$. We thus parametrize the Uber game with two exponents vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^m$ accounting for $\mathsf{pk}(\boldsymbol{x})$ and $\mathsf{pk}(\boldsymbol{y})$. The value to distinguish from random sampling using the knowledge of these public keys is then $g_T^{Q(\boldsymbol{x},\boldsymbol{y})} = \phi(\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y}))$.

Lemma 1 (Decisional Uber assumption implies nnPRF-O_ϕ). *Using the above notations, assume the decisional* $(\boldsymbol{F}, \boldsymbol{H}, \boldsymbol{K}, Q)$-Uber *assumption holds in* $\mathcal{G}$. *Then the nnPRF-*O_ϕ *assumption holds for* (ϕ-AKG, PRF).

Proof. Assuming an adversary $\mathcal{A}$ is able to win at the nnPRF-O_ϕ experiment with significant probability. We construct an adversary $\mathcal{B}$ able to break the Uber game in $\mathcal{G}$. Suppose $\mathcal{B}$ receives challenge $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, C)$. It wins if it is able to tell whether $y = g_T^{Q(\boldsymbol{x},\boldsymbol{y})}$ for some $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^m$ or $y \leftarrow\!\!\$\ \mathbb{G}_T$.

Observe that vector $\boldsymbol{X}$ can be written as a concatenation $(\boldsymbol{X}_X, \boldsymbol{X}_Y)$ where $\boldsymbol{X}_X = g_1^{\boldsymbol{F}^{\mathsf{pk}}(\boldsymbol{x})}$ and $\boldsymbol{X}_Y = g_1^{\boldsymbol{F}^{\mathsf{pk}}(\boldsymbol{y})}$ for some $\boldsymbol{x}$ and $\boldsymbol{y}$. The same applies to vectors $\boldsymbol{Y}$ and $\boldsymbol{Z}$. Adversary $\mathcal{B}$ can thus instantiates an nnPRF-O_ϕ game for $\mathcal{A}$ with $\mathsf{pk}_1 = (\boldsymbol{X}_X, \boldsymbol{Y}_X, \boldsymbol{Z}_X)$, $\mathsf{pk}_2 = (\boldsymbol{X}_Y, \boldsymbol{Y}_Y, \boldsymbol{Z}_Y)$ and challenge $y_b = \mathsf{PRF}(C)$. $\mathcal{A}$ forwards its output as a bit b'.

Adversary $\mathcal{A}$ is able to distinguish between $\mathsf{PRF}(\phi(\mathsf{pk}_1, \mathsf{sk}_2)) = \mathsf{PRF}(g_T^{Q(\boldsymbol{x},\boldsymbol{y})})$ and uniform sampling. Furthermore, uniform sampling is indistinguishable from $\mathsf{PRF}(y)$ for $y \leftarrow\!\!\$\ \mathbb{G}_T$ by definition of pseudorandom functions. Adversary $\mathcal{A}$ is thus able to distinguish $g_T^{Q(\boldsymbol{x},\boldsymbol{y})}$ from uniform sampling on $\mathbb{G}_T$. $\mathcal{B}$ only has to forward bit b' to the challenger to win with significant probability. Thus, the advantage of $\mathcal{A}$ in the nnPRF-O_ϕ experiment is bounded by the advantage of an adversary $\mathcal{B}$ against the Uber game, hence

$$\mathsf{Adv}^{\mathsf{nnPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{A}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda).$$

Hence, if the advantage on the right is negligible, it follows the nnPRF-O_ϕ assumption holds. □

4.2 Symmetric pairings

We give ϕ-AKG constructions from AKG schemes using type-1 bilinear groups. We provide a security analysis of the corresponding nnPRF-O_ϕ assumption using Lemma 1. Throughout the entire section we consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ of type 1. We write $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$ and $g := g_1 = g_2$. We assume the underlying PRFs are secure under the definitions of Appendix B. The proofs of lemmas and corollaries of this section can be found in Appendix A.

Key pairs of type (x, g^x). In the general DL context, Lemma 1 implies the following result.

Corollary 1. *Under the decisional* $((X_1, Y_1), \varnothing, \varnothing, X_1Y_1(X_1+Y_1))$-$\mathsf{Uber}$ *assumption, a* ϕ*-AKG scheme of type* $((1), \varnothing, \varnothing, \varnothing|(X_1), \varnothing, \varnothing|X_1Y_1(X_1+Y_1))$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

The assumption used in the above corollary is easily seen to imply the DBDH assumption. It is unknown to the authors whether these two assumptions are equivalent or not.

Let us now assume a trusted setup where a random generator h is available as part of the public parameter output by `AKG.Setup`. We denote by α the discrete logarithm of h in base g_1.

Corollary 2. *In this trusted setup and under the DBDH assumption, a pairing-based* ϕ*-AKG of type* $((1), \varnothing, \varnothing, \varnothing|(X_1), \varnothing, \varnothing|\alpha X_1Y_1)$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

Key pairs of type $((x_1, x_2), (g^{x_1}, g^{x_2}))$. In the case where two DL values consitute the key pairs, we can emulate h in the trusted setup above using shared generator g^{xy}.

Lemma 2. *The DBDH assumption in* $\mathcal{G}$ *implies the decisional* $((X_1, X_2, Y_1, Y_2), \varnothing, \varnothing, X_1X_2Y_1Y_2)$-$\mathsf{Uber}$ *assumption.*

Corollary 3. *Under the DBDH assumption, a pairing-based* ϕ*-AKG scheme of type* $((1,1),\varnothing,\varnothing,\varnothing|(X_1, X_2),\varnothing,\varnothing|X_1X_2Y_1Y_2)$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

4.3 Asymmetric pairings

Following the previous section, we build pairing-based ϕ-AKG schemes from AKG schemes with key structures using type 2 or 3 bilinear groups. Throughout the entire section we consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ of type 2 or 3. The proofs of lemmas and corollaries of this section can be found in Appendix A.

Key pairs of type (x, g_1^x). Unlike in the type-1 context where the DDH assumption does not hold in $\mathbb{G}_1 = \mathbb{G}_2$, here we can use the XDH assumption.

Lemma 3. *For a type-2/3 pairing, the decisional* $((X_1, Y_1), \varnothing, \varnothing, X_1Y_1)$-Uber *assumption is implied by the XDH assumption.*

Corollary 4. *Under the XDH assumption, a type-2/3 pairing-based ϕ-AKG scheme of type* $((1), \varnothing, \varnothing, \varnothing | (X_1), \varnothing, \varnothing | X_1Y_1)$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

Key pairs of type (x, g_2^x). For such public keys, care must be taken in designing ϕ for type-2 pairings as value $g_1^x = \gamma(g_2^x)$ is available via public key g_2^x. This means, for instance, that mapping $\phi(x, g_2^y) = e(g_1^x, g_2^y) = e(\gamma(g_2^x), g_2^y)$ cannot be used. This is not an issue for type-3 pairings in which γ is intractable.

Corollary 5. *Under the SXDH assumption, a type-3 pairing ϕ-AKG scheme of type* $((1), \varnothing, \varnothing, \varnothing | \varnothing, (X_1), \varnothing | X_1Y_1)$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

Key pairs of type (g_2^x, g_1^x). Now consider secret keys of type g_2^x and public keys of type g_1^x for some exponent x. A mapping ϕ can be defined via $\phi(\mathsf{sk}_1, \mathsf{pk}_2) := e(g_1^y, g_2^x)$.

Corollary 6. *Under the XDH assumption, a type-2/3 pairing-based ϕ-AKG scheme of type* $((0), \varnothing, (X_1), \varnothing | (X_1), \varnothing, \varnothing | X_1Y_1)$ *provides an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

Key pairs of type (g_1^x, g_2^x). In the opposite configuration, only type-3 pairings are usable following the same considerations as with keys of type (x, g_2^x). A mapping ϕ can be defined via $\phi(\mathsf{sk}_1, \mathsf{pk}_2) := e(g_1^x, g_2^y)$.

Corollary 7. *Under the SXDH assumption, a type-3 pairing ϕ-AKG scheme of type* $((0), \varnothing, (X_1), \varnothing | \varnothing, (X_1), \varnothing | X_1Y_1)$ *implies an ARKG scheme with PK-unlinkability and hsKS-security under the transformation in Figure 5.*

Remark 2. For type-3 pairings, independently combining any amount of the previous key types lead to a secure ARKG scheme under the SXDH assumption by taking the product of the ϕ mappings. For instance, key pairs of type $\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{x}) = (x_1, x_2, x_3, g_1^{x_4}, g_2^{x_5}), (g_1^{x_1}, g_1^{x_2}, g_2^{x_3}, g_2^{x_4}, g_1^{x_5})$ are associated with pairing

$$\phi(\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y})) = e(g_1^{y_1}, g_2^{x_1}) e(g_1^{y_2}, g_2^{x_2}) e(g_1^{x_3}, g_2^{y_3}) e(g_1^{y_4}, g_2^{x_4}) e(g_1^{x_5}, g_2^{y_5}).$$

Since SXDH implies DDH in $\mathbb{G}_T$, using elements from the target group in the public key is also possible and only requires more taking the corresponding products in the definition of ϕ.

5 Applications to Pairing-based Signatures

We now turn our attention to selected pairing-based signature schemes across different types of bilinear groups to illustrate the application of our general transformation to ARKG. It must be noted that the security of the signature scheme is independent from the security of the ARKG scheme, as highlighted by the composability results in [1]. We take common signature schemes and extract the key generation steps from them to build AKG schemes outputting the same structure of keys. These AKG schemes are assumed to provide key secrecy, however we also consider in some cases subsets of the signature scheme's key structure. The hardness of the DL assumption in the public key group will imply key secrecy in these cases.

For each scheme we indicate the structure of the original private-public key pair and the corresponding ϕ-mapping that is used to compute the derived private-public key pair via the ARKG transformation. For simplicity, in our descriptions we focus only on the structures of the key pairs without repeating the remaining computation steps behind the transformation in Figure 5 such as computation of KDFs or MACs that are also part of the transformation. In the following, whenever a scheme requires a public key to contain randomly sampled generators, we assume these to be generated as part of the public parameters.

5.1 ARKG for selected signature schemes over type-1 pairings

In this section we use the notations of Section 4.2. We consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ of type 1 where $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$ and $g := g_1 = g_2$.

BLS-1 signatures [7]. The Boneh, Lynn, and Shacham pairing-based signature scheme is one of the most famous and simple pairing-based schemes. Recall that the key generation algorithm `KGen` for BLS-1 signatures outputs keys of the form $(\mathsf{sk}, \mathsf{pk}) = (x, g^x)$ for some $x \leftarrow\!\!\$\ \mathbb{Z}_p$. The underlying `AKG` scheme is derived from the key generation steps of BLS. $\texttt{AKG.Setup}(1^\lambda)$ returns $\mathcal{G}$, $\texttt{AKG.SKGen}(\mathsf{pp})$ computes $\mathsf{sk} = x \leftarrow\!\!\$\ \mathbb{Z}_p$, $\texttt{AKG.PKGen}(\mathsf{pp}, \mathsf{sk})$ computes $\mathsf{pk} = g^{\mathsf{sk}}$ and $\texttt{AKG.Check}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$ verifies that $g^{\mathsf{sk}} = \mathsf{pk}$. Let us first extend this AKG scheme to a ϕ-AKG scheme in a trusted setup.

Trusted setup. Suppose an additional trusted generator h of $\mathbb{G}$ is available and define mapping $\phi : (x, g^y) \mapsto e(g^y, h^x)$. This extends the aforementionned AKG scheme to a ϕ-AKG scheme. Corollary 2 of Section 4.2 then implies that under the DBDH assumption, the transformation of Figure 5 yields an ARKG instance compatible with the BLS key structure that has the PK-unlinkability and hsKS-security properties.

Trustless setup. We can avoid the trusted setup by using, instead of the underlying AKG scheme directly derived from the signature scheme, the following key structure: $(\mathsf{sk}, \mathsf{pk}) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$ and mapping

$$\phi : ((x_1, x_2), (g^{y_1}, g^{y_2})) \mapsto e(g^{y_1}, (g^{x_2} g^{y_2})^{x_1}))$$

where public element $h = g^{x_2} g^{y_2}$ acts as a trusted generator. Algorithms `SKGen`, `PKGen` and `Check` need to be modified accordingly. Assuming $x_2 + y_2$ is not known to anyone, the AKG provides key secrecy. Since the additional key elements are equivalent to a shared trusted generator h, the security of the resulting ARKG instance is also implied by the DBDH assumption.

Alternatively, under the assumption of Corollary 1, one can avoid doubling keys and use pairing

$$\phi : (x, g^y) \mapsto e((g^x g^y)^x, g^y).$$

CL signatures [9]. The Camenisch-Lysyanskaya signature scheme is based on the LRSW assumption introduced in [16] and can be used in anonymous credential systems and group signature schemes. The key structure of the underlying AKG scheme is $(\mathsf{sk}, \mathsf{pk}) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$ for some $(x_1, x_2) \leftarrow\!\!\$\ \mathbb{Z}_p^2$. As such, we can extend it to a ϕ-AKG scheme via mapping $\phi : ((x_1, x_2), (g^{y_1}, g^{y_2})) \mapsto e(g^{y_1}, g^{y_2})^{x_1 x_2}$. Applying Corollary 3, we deduce that the transformation of Figure 5 yields an ARKG instance compatible with the CL key structure that has the PK-unlinkability and hsKS-security properties under the DBDH assumption.

5.2 ARKG for selected signature schemes over type-3 pairings

In this section we use the notations of Section 4.3 and consider a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ of type 3.

Let $l, s \geq 0$ and consider signature schemes using key pairs of the form

$$(\mathsf{sk}, \mathsf{pk}) = ((x_1, \cdots, x_{l+s}), (g_1^{x_1}, \cdots, g_1^{x_l}, g_2^{x_{l+1}}, \cdots, g_2^{x_{l+s}})) \in \mathbb{Z}_p^{l+s} \times (\mathbb{G}_1^l \times \mathbb{G}_2^s).$$

We transform the underlying AKG into a ϕ-AKG scheme using mapping

$$\phi : (\mathsf{sk}(\boldsymbol{x}), \mathsf{pk}(\boldsymbol{y}))) \mapsto \prod_{i=1}^{l} e(g_1^{y_i}, g_2^{x_i}) \prod_{i=1}^{s} e(g_1^{x_{l+i}}, g_2^{y_{l+i}}).$$

The security of the associated ARKG scheme follows from the SXDH assumption as mentioned in Remark 2. The key structures used in the following three signature schemes can therefore be used in a secure ARKG scheme under the SXDH assumption.

BLS-3 signatures [17] BLS-3 signatures are defined in [17] and use keys of the form $(\mathsf{sk}, \mathsf{pk}) = (x_1, g_2^{x_1})$ for some $x_1 \leftarrow\!\!\$\ \mathbb{Z}_p$.

Pointcheval-Sanders signatures [11] Pointcheval-Sanders signatures defined in [11] use type-3 pairings to obtain all the advantages of CL signatures with shorter signature elements and more efficient algorithms. The key structure of the underlying AKG scheme is $(\mathsf{sk}, \mathsf{pk}) = ((x_1, x_2), (g_1^{x_1}, g_1^{x_2}))$ for some $(x_1, x_2) \leftarrow\$ \mathbb{Z}_p^2$.

SPS-EQ [10]**.** Structure-preserving signatures on equivalence classes use type-3 pairings and keys of the form $(\mathsf{sk}, \mathsf{pk}) = ((x_1, \cdots, x_l), (g_2^{x_1}, \cdots, g_2^{x_l}))$.

Waters signatures [8]**.** Waters signatures exist in all three types of bilinear groups and can be tuned either to have very short signatures or very short shared hash function parameters. We consider the type-3 setting. The key structure for this scheme is $(\mathsf{sk}, \mathsf{pk}) = ((g_1^{x_1}, x_2, \cdots, x_l), (g_T^{x_1}, g_2^{x_2}, \cdots, g_2^{x_l}))$. Recall that ARKG requires an AKG scheme generating such key pairs with two independent algorithms `AKG.SKGen` and `AKG.PKGen`. With `AKG.SKGen` outputting secret keys of the form $(g_1^{x_1}, x_2, \cdots, x_l)$, we obtain an algorithm outputting a corresponding public key of the form $(g_T^{x_1}, g_2^{x_2}, \cdots, g_2^{x_l})$ by computing $g_T^{x_1}$ as $e(g_1^{x_1}, g_2)$. A composite mapping for this AKG scheme is

$$(g_1^{x_1}, x_2, \cdots, x_l), (g_T^{y_1}, g_2^{y_2}, \cdots, g_2^{y_l}) \mapsto (g_T^{y_1})^{x_2} e(g_1^{x_1}, g_2^{y_2}) e(g_1^{x_2}, g_2^{y_2}) \cdots e(g_1^{x_l}, g_2^{y_l}).$$

The security of the ARKG instance obtained from this scheme is the decisional $((X_2, \cdots, X_l, Y_2, \cdots, Y_l), \varnothing, (X_1, Y_1), X_2Y_1 + X_1Y_2 + X_2Y_2 + \cdots X_lY_l)$-Uber assumption. As with the previous schemes, it is implied by the SXDH assumption.

5.3 Implementation and Performance

In Table 1, the mean time (in milliseconds) of ten invocations of ARKG is presented, taken from our reference implementation[4] without any optimisation. The benchmarking was performed on an Intel i5-6600, with a clock speed of 3.30GHz. When combined, `DerivePK` and `DeriveSK` give the total time required to generate a derived key pair, which is also presented in the table. The schemes based on type-1 pairings (BLS, CL) are implemented in C using the PBC[5] and Sodium [6] libraries. The type-2/3 schemes (PS, SPS-EQ, Waters) use the readily-available bplib package in Python. For our implementations in C, which are much more performant, only an increase of 3.5ms and 4.0ms is seen when comparing ARKG-derived keys to the underlying AKG `KeyGen` algorithm, for BL- and CL-compatible keys, respectively. However, our Python implementations, which are not optimised in any way, take the order of 100ms. This is primarily due to the many group operations required for `DerivePK`.

[4] https://gitlab.surrey.ac.uk/sccs/bp-arkg
[5] https://crypto.stanford.edu/pbc/
[6] https://github.com/jedisct1/libsodium

Table 1: Mean time in milliseconds for each ARKG algorithm, along with the respective `AKG.KGen`. BLS-1/3 and CL are written in C, PS, SPS-EQ and Waters are implemented in python.

	`DerivePK`	`DeriveSK`	`Check`	`ARKG` total	`AKG.KGen`
BLS-1 [7]	3.56	1.07	0.63	5.26	0.63
BLS-3 [17]	2.92	0.99	0.62	4.53	0.61
CL [9]	5.36	0.89	2.21	6.26	2.24
PS [11]	99.23	8.29	0.89	107.52	0.94
SPS-EQ [10]	123.34	17.13	10.89	140.47	5.62
Waters [8]	127.40	17.12	11.52	144.52	8.96

6 Conclusion

In this work we proposed a general approach for constructing Asynchronous Remote Key Generation (ARKG) from a simpler building block, which we call Asymmetric Key Generation (AKG) and its extension ϕ-AKG. Through an appropriate choice of the mapping ϕ and the underlying private/public key groups for AKG we were able to generalise the first ARKG scheme for DL-based key pairs from [1] and, more importantly, obtain first ARKG constructions catering for different types of bilinear groups and key structures commonly used in pairing-based cryptography. Specifically, our general transformation from pairing-based ϕ-AKG to ARKG allows to generate key pairs whose security is implied by a family of decisional Uber assumption [12], and hence all assumptions that imply the latter, including DBDH, XDH, SXDH. To demontrate the power of our general approach we provide concrete pairing-based ϕ-AKG instances for different key pairs and types of pairings, and illustrate their use for some well-known pairing-based signature schemes. Our work is supported by appropriate publicly-accessible implementations and benchmarks, showing that the overhead for generating key pairs using ARKG based on our transformation introduces only a neglible overhead (i.e., an additional 3.5ms in our most performant implementation) when compared to the original key generation algorithms of the respective schemes.

References

1. Nick Frymann, Daniel Gardham, Franziskus Kiefer, Emil Lundberg, Mark Manulis, and Dain Nilsson. Asynchronous remote key generation: An analysis of yubico's proposal for w3c webauthn. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 939–954, New York, NY, USA, 2020. Association for Computing Machinery.
2. Nick Frymann, Daniel Gardham, and Mark Manulis. Asynchronous remote key generation for post-quantum cryptosystems from lattices. Cryptology ePrint Archive, Paper 2023/419, 2023. https://eprint.iacr.org/2023/419.

3. Dirk Balfanz, Alexei Czeskis, Jeff Hodges, J.C. Jones, Michael B. Jones, Akshay Kumar, Angelo Liao, Rolf Lindemann, and Emil Lundberg. Web authentication: An API for accessing public key credentials level 1. Technical report, 2019.
4. Peter Todd. Stealth addresses, 2014. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html.
5. Nicolas van Saberhagen. Cryptonote v2.0, 2013. https://cryptonote.org/whitepaper.pdf.
6. Nick Frymann, Daniel Gardham, and Mark Manulis. Unlinkable delegation of webauthn credentials. In *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part III*, page 125–144, Berlin, Heidelberg, 2022. Springer-Verlag.
7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
8. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 114–127, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
9. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
10. Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 491–511, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
11. David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610*, page 111–126, Berlin, Heidelberg, 2016. Springer-Verlag.
12. Xavier Boyen. The uber-assumption family. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, pages 39–56, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
13. Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. Prf-odh: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 651–681, Cham, 2017. Springer International Publishing.
14. Lior Rotem. Revisiting the uber assumption in the algebraic group model: Fine-grained bounds in hidden-order groups and improved reductions in bilinear groups. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
15. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 121–151, Cham, 2020. Springer International Publishing.
16. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, pages 184–199, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
17. Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. Cryptology ePrint Archive, Paper 2009/060, 2009. https://eprint.iacr.org/2009/060.

Appendix

A More proofs

A.1 Proof of Theorem 3

Define $\mathcal{G}_0$ as the starting experiment $\mathsf{Exp}^{\mathsf{msKS}}_{\mathcal{A}}$. As such,

$$\Pr\left[\mathsf{Exp}^{\mathsf{msKS}}_{\mathcal{A}} = 1\right] = \Pr[\mathcal{G}_0 = 1].$$

Define $\mathcal{G}_1$ as $\mathcal{G}_0$ except,

- During oracle calls to $\mathsf{O}_{\mathsf{pk}'}$, line 4 of `DerivePK` is replaced with

$$r \leftarrow\!\$\ \mathbb{G}_{\mathsf{sk}};\quad P \leftarrow \texttt{AKG.PKGen}(\mathsf{pp}, ck) \cdot \texttt{AKG.PKGen}(\mathsf{pp}, r).$$

  An internal list List of elements (E, e, r) generated during the calls is maintained by the challenger.
- If $\mathcal{A}$ queries $\mathsf{O}_{\mathsf{sk}'}$ with $E \in \mathsf{cred}$ such that $E \in \mathsf{List}$, then line 4 of `DeriveSK` is replaced with "**return** $\mathsf{sk}' = \texttt{AKG.SKCombine}(\mathsf{pp}, ck, r)$". This ensures the validity of derived keys.

Games $\mathcal{G}_0$ and $\mathcal{G}_1$ are indistinguisable. Indeed, suppose an adversary $\mathcal{C}$ is able to distinguish between both games with non-negligible probability. $\mathcal{C}$ is thus able to differentiate between the original ($\mathcal{G}_0$) and modified ($\mathcal{G}_1$) versions of the `DerivePK` algorithm. We construct an adversary $\mathcal{D}$ able to break the $\mathsf{Exp}^{\mathsf{OTB}}_{\mathbb{G}_{\mathsf{pk}},\mathcal{A}}$ experiment.

Adversary $\mathcal{D}$ receives challenge (z_b, x) and wins if it correctly guesses bit b where $z_0 \leftarrow x \cdot z$ and $z_1 \leftarrow y \cdot z$ for uniformly sampled $(x, y, z) \in \mathbb{G}_{\mathsf{pk}}$ and $b \leftarrow\!\$\ \{0, 1\}$. Adversary $\mathcal{D}$ sets up game a game where it asks $\mathcal{C}$ which version of `DerivePK` was used to produce output $P = y_b$ on input $(\mathsf{pp}, S, aux) = (\bot, x, \bot)$ The distributions $\texttt{AKG.PKGen}(\mathsf{pp}, ck)$ and z are indistinguishable as the ϕ-AKG scheme provides uniform sampling of keys. The same goes for distributions $\texttt{AKG.PKGen}(\mathsf{pp}, r)$ and y. As such, $\mathcal{C}$ answers correctly to the challenge with significant probability and $\mathcal{D}$ breaks the $\mathsf{Exp}^{\mathsf{OTB}}_{\mathbb{G}_{\mathsf{pk}},\mathcal{A}}$ experiment by forwarding the answer. Thus

$$\Pr[\mathcal{G}_0 = 1] = \Pr[\mathcal{G}_1 = 1].$$

Assume an adversary $\mathcal{A}$ is able to win the msKS experiment with non-negligible probability. We will construct an adversary $\mathcal{B}$ able to break the snPRF-O_ϕ experiment.

Adversary $\mathcal{B}$ receives a snPRF-O_ϕ challenge (y_b, x) and public parameters $(\mathsf{bp}, \mathsf{pk}_1, \mathsf{pk}_2)$. It sets up game $\mathcal{G}_1$ with $\mathsf{sk} \leftarrow\!\bot$ and $\mathsf{pk} \leftarrow \mathsf{pk}_1$ and uses x for the label of KDF_1. It then challenges $\mathcal{A}$ create a forgery on y_b and answers its oracle calls honestly. Adversary eventually $\mathcal{A}$ answers with a triple $(\mathsf{sk}^\star, \mathsf{pk}^\star, \mathsf{cred}^\star)$. Adversary $\mathcal{B}$ extracts E from $\mathsf{cred}^\star$ and uses a single query to the O_ϕ oracle to get $ck \leftarrow \mathsf{KDF}_1(\phi(E, \mathsf{sk}_1))$ where sk_1 is the secret key associated to pk_1. $\mathcal{B}$ can

then compute this secret key as $\mathsf{sk}_1 = \mathtt{AKG.SKInv}(\mathsf{pp}, \mathsf{sk}^\star, ck)$ and compare y_b and $\mathsf{KDF}_1(\phi(\mathsf{pk}_2, \mathsf{sk}_1))$. It returns 0 if they are equal, otherwise 1.

A query from $\mathcal{B}$ to O_ϕ with $E = \mathsf{pk}_2$ aborts the experiment. This happens with probability $\epsilon = q/n$ where $n = \#\mathbb{G}_{\mathsf{pk}}$ by the UPK assumption where q is the number of oracle queries made by $\mathcal{A}$. It becomes negligible when n is large.

Thus, the advantage of $\mathcal{A}$ in msKS is bounded by the advantage of an adversary $\mathcal{B}$ against the snPRF-O_ϕ game, hence

$$\mathsf{Adv}^{\mathsf{msKS}}_{\mathcal{A}}(\lambda) \leqslant (1-\varepsilon)\mathsf{Adv}^{\mathsf{snPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{B}}(\lambda).$$

By assumption the $\mathsf{Adv}^{\mathsf{snPRF\text{-}O}_\phi}_{\mathsf{PRF},\mathcal{B}}(\lambda)$ experiment is hard, it follows that this compiled ARKG scheme is msKS-secure. □

In the following proofs of lemmata and corollaries, we omit sets used to define decisional Uber experiments when they are empty for brevity.

A.2 Proof of Lemma 2

Assume an adversary $\mathcal{A}$ is able to win at the Uber experiment with significant probability. We construct an adversary $\mathcal{B}$ able to break the DBDH game. Suppose $\mathcal{B}$ receives challenge $(g^x, g^y, g^z, C) \in \mathbb{G}^3 \times \mathbb{G}_T$. It wins if it is able to tell whether $C = e(g,g)^{xyz}$ or $C \leftarrow\!\!\$\ \mathbb{G}_T$.

Adversary $\mathcal{B}$ instantiates a Uber game for $\mathcal{A}$ with $\boldsymbol{X} = (g^x, g^y, g^z, g)$ and challenge C. $\mathcal{A}$ forwards its output as a bit b'.

Adversary $\mathcal{A}$ is by definition able to distinguish $g_T^{Q(x,y,z,1)} = g_T^{xyz}$ from uniform sampling in $\mathbb{G}_T$. $\mathcal{B}$ only has to forward bit b' to the challenger to win with significant probability.

Thus, the advantage of $\mathcal{A}$ in the Uber experiment is bounded by the advantage of an adversary $\mathcal{B}$ against the DBDH game, hence

$$\mathsf{Adv}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{G},\mathcal{A}}(\lambda).$$

By assumption the DBDH experiment is hard, it follows that the Decisional Uber assumption holds. □

A.3 Proof of Lemma 3

Assume an adversary $\mathcal{A}$ is able to win at the Uber experiment with significant probability. We construct an adversary $\mathcal{B}$ able to break the DDH game in $\mathbb{G}_1$. Suppose $\mathcal{B}$ receives challenge $(g_1^x, g_1^y, C) \in \mathbb{G}_1^3$. It wins if it is able to tell whether $C = g_1^{xy}$ or $C \leftarrow\!\!\$\ \mathbb{G}_1$.

Adversary $\mathcal{B}$ instantiates a Uber game for $\mathcal{A}$ with $\boldsymbol{X} = (g_1^x, g_1^y)$ and challenge C. $\mathcal{A}$ forwards its output as a bit b'.

Adversary $\mathcal{A}$ is by definition able to distinguish $g_T^{Q(x,y)} = g_T^{xy}$ from uniform sampling in $\mathbb{G}_T$. As such, $\mathcal{A}$ is able to distinguish xy from random sampling in $\mathbb{Z}_p$ and thus g_1^{xy} from random sampling in $\mathbb{G}_1$. $\mathcal{B}$ only has to forward bit b' to the challenger to win with significant probability.

Thus, the advantage of $\mathcal{A}$ in the Uber experiment is bounded by the advantage of an adversary $\mathcal{B}$ against the DDH game, hence

$$\mathsf{Adv}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda).$$

By assumption the DDH experiment is hard, it follows that the decisional Uber assumption holds. □

A.4 Proof of Corollary 3

By Lemma 2, the Decisional $((X_1, X_2, Y_1, Y_2), \varnothing, \varnothing, X_1X_2Y_1Y_2)$-Uber assumption is implied by the DBDH assumption. As such, it follows from Lemma 1 that the nnPRF-O_ϕ assumption holds for the ϕ-AKG scheme. We can thus conclude by Theorem 1 and Theorem 2 . □

A.5 Proof of Corollary 2

Denote by A_1 the Decisional $((X_1, X_2, Y_1, Y_2), \varnothing, \varnothing, X_1X_2Y_1Y_2)$-Uber assumption and by A_2 the Decisional $((X_1, Y_1), \varnothing, \varnothing, \alpha X_1X_2)$-Uber assumption. According to Lemma 2, it suffices to show that $A_1 \Rightarrow A_2$ since $\mathsf{DBDH} \Rightarrow A_1$.

Assume an adversary $\mathcal{A}$ is able to win at the A_2 experiment with significant probability. We construct an adversary $\mathcal{B}$ able to break the A_1 game. Suppose $\mathcal{B}$ receives challenge set $\boldsymbol{X} = (g^x, g^y)$ and challenge C. It wins if it is able to tell whether $C = e(g, g)^{\alpha xy}$ or $C \leftarrow\!\$\ \mathbb{G}_T$.

Adversary $\mathcal{B}$ instantiates an A_1 game for $\mathcal{A}$ with input set $\boldsymbol{X} = (g^x, h, g^y, g)$ and challenge C. $\mathcal{A}$ forwards its output as a bit b'.

Adversary $\mathcal{A}$ is by definition able to distinguish $g_T^{Q(x,\alpha,y,1)} = g_T^{\alpha xy}$ from uniform sampling in $\mathbb{G}_T$. $\mathcal{B}$ only has to forward bit b' to the challenger to win with non-negligible probability.

Thus, the advantage of $\mathcal{A}$ in the Uber experiment is bounded by the advantage of an adversary $\mathcal{B}$ against the DBDH game, hence

$$\mathsf{Adv}^{\mathsf{Uber}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{G},\mathcal{A}}(\lambda)$$

and therefore we conclude that $A_1 \Rightarrow A_2$. □

A.6 Proof of Corollary 4

By Lemma 3, the Decisional $((X_1, Y_1), \varnothing, \varnothing, X_1Y_1)$-Uber assumption is implied by the XDH assumption. As such, it follows from Lemma 1 that the nnPRF-O_ϕ assumption holds for the ϕ-AKG scheme. We can thus conclude by Theorem 1 and Theorem 2 . □

A.7 Proof of Corollary 5

The proof is similar to that of Corollary 4 except group $\mathbb{G}_2$ (where the DDH assumption holds) is used in place of $\mathbb{G}_1$.

A.8 Proof of Corollary 6

The proof is similar to that of Corollary 4 as the public key part $((X_1), \varnothing, \varnothing)$, polynomial $Q = X_1 Y_1$ and the assumptions are identical.

A.9 Proof of Corollary 7

The proof is similar to that of Corollary 5 since the public key part $(\varnothing, (X_1), \varnothing)$, polynomial $Q = X_1 Y_1$ and the assumptions are identical.

B Preliminaries

In this section we recall general definitions such as pseudorandom functions, key derivation functions and message authentication codes.

Definition 12 (Pseudorandom function). *A pseudorandom function (PRF)* $\mathsf{PRF} : K \times M \to M'$ *takes as input a key* k *and a message* m*. It outputs a new message* m' *not necessarily from the same space as* m*. Define oracle* $\mathsf{O}_{\mathsf{PRF}}$ *parametrized by* k *and taking as input a message* $m' \neq m$*. It outputs either* $\mathsf{PRF}(k, m')$ *or* $f(m')$ *where* f *is a truly random function. A PRF is secure if the following advantage is negligible in* 1^λ *for all PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{RAND}}_{\mathsf{PRF},\mathcal{A}} = 1\right] - \frac{1}{2} \right|,$$

where the output randomness RAND *experiment is defined in Figure 8.*

Definition 13 (Key Derivation Function). *A key derivation function* $\mathsf{KDF} : K \times L \to K'$ *takes as input a key* k *and a label* l*. It outputs a new key* k' *not necessarily from the same keyspace as* k*. A KDF is secure if the following advantage is negligible in* 1^λ *for all PPT adversary* $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{KDF}}_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{IND}}_{\mathsf{KDF},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$

where the indistinguishability IND *experiment is defined in figure 8. In the following we will fix once and for all a KDF function* $\mathsf{KDF} : \mathbb{G}_{\mathsf{pk}} \times L \to \mathbb{G}_{\mathsf{sk}}$*, two labels* l_1*,* l_2 *and consider* $\mathsf{KDF}_1 = \mathsf{KDF}(\cdot, l_1)$ *and* $\mathsf{KDF}_2 = \mathsf{KDF}(\cdot, l_2)$

Definition 14 (Message Authentication Code). *A Message Authentification Code (MAC) is a triple* $\mathsf{MAC}=($`KGen`, `Tag`, `Verify`$)$*.* $\mathtt{KGen}(1^\lambda)$ *takes as input a security parameter* 1^λ *and outputs a secret key* $mk \leftarrow\!\!\$\ \{0,1\}^\lambda$*,* $\mathtt{Tag}(mk, m)$ *outputs a tag* μ *for a secret key* mk *and a message* m *and* $\mathtt{Verify}(mk, m, \mu)$ *outputs* 1 *if the tag* μ *is valid for* mk *and* m*, otherwise* 0*. A MAC scheme is correct if for every* $\lambda \in \mathbb{N}, m \in \{0,1\}^*$*,*

$$(mk \leftarrow \mathtt{KGen}(1^\lambda);\ \ \mu \leftarrow \mathtt{Tag}(mk, m)) \Rightarrow \mathtt{Verify}(mk, m, \mu),$$

Define oracle $\mathsf{O}_{\mathsf{Tag},mk}$ *parametrized by a key* mk *and taking a message* m *as input. It returns the result of* $\mathtt{Tag}(mk, m)$*. A MAC is unforgeable if the following advantage is negligible in* λ *for all PPT adversary* $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{UNF}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = \Pr\Big[\mathsf{Exp}^{\mathsf{UNF}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = 1\Big],$$

where the unforgeability UNF *experiment is defined in Figure 8.*

$\mathsf{Exp}^{\mathsf{RAND}}_{\mathsf{PRF},\mathcal{A}}$

1 : $k \leftarrow\!\!\$\ K$
2 : $m \leftarrow\!\!\$\ M$
3 : $y_0 \leftarrow \mathsf{PRF}(k, m)$
4 : $y_1 \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
5 : $b \leftarrow\!\!\$\ \{0,1\}$
6 : $b' \rightarrow \mathcal{A}^{\mathsf{O}_{\mathsf{PRF}}}(y_b)$
7 : **return** $b' \stackrel{?}{=} b$

$\mathsf{Exp}^{\mathsf{IND}}_{\mathsf{KDF},\mathcal{A}}$

1 : $k \leftarrow\!\!\$\ K$
2 : $l \leftarrow\!\!\$\ L$
3 : $y_0 \leftarrow \mathsf{KDF}(k, l)$
4 : $y_1 \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
5 : $b \leftarrow\!\!\$\ \{0,1\}$
6 : $b' \rightarrow \mathcal{A}(y_b)$
7 : **return** $b' \stackrel{?}{=} b$

$\mathsf{Exp}^{\mathsf{UNF}}_{\mathsf{MAC},\mathcal{A}}$

1 : $mk \leftarrow \mathtt{KGen}(1^{\lambda})$
2 : $(m^{\star}, \mu^{\star}) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Tag},mk}}$
3 : **return** $m \neq m^{\star}$
4 : $\wedge\ \mathtt{Verify}(mk, m^{\star}, \mu^{\star})$

Fig. 8: The security experiments associated to the PRF, KDF and MAC definitions.