Reconfigurable Cryptography: A flexible approach to long-term security*

Julia Hesse and Dennis Hofheinz and Andy Rupp†

Karlsruhe Institute of Technology, Germany

{julia.hesse, dennis.hofheinz, andy.rupp}@kit.edu

October 28, 2015

Abstract

We put forward the concept of a *reconfigurable* cryptosystem. Intuitively, a reconfigurable cryptosystem allows to increase the security of the system at runtime, by changing a single central parameter we call common reference string (CRS). In particular, e.g., a cryptanalytic advance does not necessarily entail a full update of a large public-key infrastructure; only the CRS needs to be updated. In this paper we focus on the reconfigurability of encryption and signature schemes, but we believe that this concept and the developed techniques can also be applied to other kind of cryptosystems.

Besides a security definition, we offer two reconfigurable encryption schemes, and one reconfigurable signature scheme. Our first reconfigurable encryption scheme uses indistinguishability obfuscation (however only in the CRS) to adaptively derive short-term keys from long-term keys. The security of long-term keys can be based on a one-way function, and the security of both the indistinguishability obfuscation and the actual encryption scheme can be increased on-the-fly, by changing the CRS. We stress that our scheme remains secure even if previous short-term secret keys are leaked.

Our second reconfigurable encryption scheme has a similar structure (and similar security properties), but relies on a pairing-friendly group instead of obfuscation. Its security is based on the recently introduced hierarchy of k-SCasc assumptions. Similar to the k-Linear assumption, it is known that k-SCasc implies $(k+1)$-SCasc, and that this implication is proper in the generic group model. Our system allows to increase k on-the-fly, just by changing the CRS. In that sense, security can be increased without changing any long-term keys.

We also offer a reconfigurable signature scheme based on the same hierarchy of assumptions.

Keywords: long-term security, security definitions, public-key cryptography.

1 Introduction

Motivation. Public-key cryptography plays an essential role in security and privacy in wide networks such as the internet. Secure channels are usually established using hybrid encryption, where the exchange of session keys for fast symmetric encryption algorithms relies on a public key infrastructure (PKI). These PKIs incorporate public keys from large groups of users. For instance, the PKI used by OpenPGP for encrypting and signing emails consists of roughly four million public keys. This PKI is continuously growing, especially so since the Snowden leaks multiplied the amount of newly registered public keys.

*This work will appear in the proceedings of TCC 2016.

†The authors were supported by DFG grants GZ HO 4534/2-2 and GZ HO 4534/4-1

One drawback of large PKIs is that they are slow to react to security incidents. For instance, consider a PKI that predominantly stores 2048-bit RSA keys, and imagine a sudden cryptanalytic advance that renders 2048-bit RSA keys insecure. In order to change all keys to, say, 4096-bit keys, every user would have to generate new keypairs and register the new public key. Similarly, expensive key refresh processes are necessary in case, e.g., a widely deployed piece of encryption software turns out to leak secret keys, the assumed adversarial resources the system should protect from suddenly increase (e.g., from the computing resources of a small group of hackers to that of an intelligence agency), etc.

In this paper, we consider a scenario where key updates are triggered by a central authority for all users/devices participating in a PKI (and not by the individuals themselves), e.g., such as a large company maintaining a PKI for its employees who wants the employees to update their keys every year or when new recommendations on minimal key lengths are released. Other conceivable examples include operators of a PKI for wireless-sensor networks or for other IoT devices. We do not consider the problem of making individually initiated key updates more efficient.

Reconfigurable Cryptography. This paper introduces the concept of reconfigurable cryptography. In a nutshell, in a reconfigurable cryptographic scheme, there are long-term and short-term public and secret keys. Long-term public and secret keys are generated once for each user, and the long-term public key is publicized, e.g., in a PKI. Using a central and public piece of information (the common reference string or CRS), long-term keys allow to derive short-term keys, which are then used to perform the actual operation. If the short-term keys become insecure (or leak), only the central CRS (but not the long-term keys) needs to be updated (and certified). Note that the long-term secret keys are only needed for the process of deriving new short-term secret keys and not for the actual decryption process. Thus, they can be kept "offline" at a secure place.

We call the process of updating the CRS *reconfiguration*. An attack model for a reconfigurable cryptography scheme is given by an adversary who can ask for short-term secret keys derived from the PKI and any deprecated CRSs. After that, the adversary is challenged on a fresh short-term key pair. This models the fact that short-term key pairs should not reveal any information about the long-term secret keys of the PKI and thus, after their leakage, the whole system can be rescued by updating only the central CRS. Note that for most such schemes (except some trivial ones described below), the entity setting up the CRS needs to be trusted not to keep a trapdoor allowing to derive short-term secret keys for all users and security levels. In order to mitigate this risk however, a CRS could also be computed in a distributed fashion using MPC techniques.

Related concepts and first examples. An objection to our approach that might come to mind when first thinking about long-term secure encryption is the following: why do we not follow a much simpler approach like letting users exchange sufficiently long symmetric encryption keys once (which allow for fast encryption/decryption), using a (slow) public key scheme with comparable security? Unfortunately, it quickly turns out that there are multiple drawbacks with this approach: advanced encryption features known only for public-key encryption (e.g., homomorphic encryption) are excluded; each user needs to maintain a secure database containing the shared symmetric keys with his communication partners; the long-term secret key of the PKE scheme needs to be kept "online" in order to be able to decrypt symmetric keys from new communication partners, etc. Hence, we do not consider this a satisfying approach to long-term security.

A first attempt to create a scheme which better complies with our concept of reconfigurable encryption could be the following: simply define the long-term keys as a sequence of short-term keys. For instance, a long-term public key could consist of RSA keys of different lengths, say, of 2048, 4096, and 8192 bits. The CRS could be an index that selects which key (or, keylength) to use as a short-term key. If a keylength must be considered broken, simply take the next. This approach is perfectly viable, but does not scale well: only an a-priori fixed number (and type) of keys can be

stored in a long-term key, and the size of such a long-term key grows linearly in the number of possible short-term keys.

A second attempt might be to use identity-based techniques: for instance, the long-term public and secret key of a user of a reconfigurable encryption scheme could be the master public and secret key of an identity-based encryption (IBE [21, 17, 6]) scheme. The CRS selects an IBE identity (used by all users), and the short-term secret key is the IBE user secret key for the identity specified by the CRS. Encryptions are always performed to the current identity (as specified by the CRS), such that the short-term secret key can be used to decrypt. In case (some of) the current short-term secret keys are revealed, simply change the identity specified in the CRS. This scheme scales much better to large numbers of reconfigurations than the trivial scheme above. Yet, security does not increase after a reconfiguration. (For instance, unlike in the trivial example above, there is no obvious way to increase keylengths through reconfiguration.)

Finally, we note that our security requirements are somewhat orthogonal to the ones found in forward security [10, 4, 9]. Namely, in a forward-secure scheme, we would achieve that revealing a current (short-term) secret key does not harm the security of *previous* instances of the scheme. In contrast, we would like to achieve that revealing the current (and previous) short-term secret keys does not harm the security of *future* instances of the scheme. Furthermore, we are interested in *increasing* the security of the scheme gradually, through reconfigurations (perhaps at the cost of decreased efficiency).

Our contribution. We introduce the concept of reconfigurable cryptography. For this purpose, it is necessary to give a security definition for a cryptographic scheme defined in *two* security parameters. This definition needs to capture the property that security can be increased by varying the short-term security parameter. As it turns out, finding a reasonable definition which captures our notion and is satisfiable at the same time is highly non-trivial. Ultimately, here we present a non-uniform security definition based on an asymptotic version of *concrete security* introduced by Bellare et al. in [3, 2]. The given definition is intuitive and leads to relatively simple proofs. Consequently, also our building blocks need to be secure against non-uniform adversaries (what can be assumed when building on non-uniform complexity assumptions). Alternatively, also a uniform security definition is conceivable which, however, would lead to more intricate proofs.

Besides a security definition, we offer three constructions: two reconfigurable public-key encryption schemes (one based on indistinguishability obfuscation [1, 12, 20], the other based on the family of SCasc assumptions [11] in pairing-friendly groups), and a reconfigurable signature scheme based on arbitrary families of matrix assumptions (also in pairing-friendly groups).

To get a taste of our solutions, we now sketch our schemes.

Some notation. We call $\lambda \in \mathbb{N}$ the long-term security parameter, and $k \in \mathbb{N}$ the short-term security parameter. λ has to be fixed at setup time, and intuitively determines how hard it should be to retrieve the long-term secret key from the long-term public key. (As such, λ gives an an upper bound of the security of the whole system. In particular, we should be interested in systems in which breaking the long-term public key should be qualitatively harder than breaking short-term keys.) In contrast, k can (and should) increase with each reconfiguration. Intuitively, a larger value of k should make it harder to retrieve short-term keys.

Our obfuscation-based reconfigurable encryption scheme. Our first scheme uses indistinguishability obfuscation [1, 12, 20], a pseudorandom generator PRG, and an arbitrary public-key encryption scheme PKE. As a long-term secret key, we use a value $x \in \{0,1\}^\lambda$; the long-term public key is $\mathsf{PRG}(x)$. A CRS consists of the obfuscation of an algorithm Gen, that inputs either a long-term public key $\mathsf{PRG}(x)$ or a long-term secret key x, and proceeds as follows:

- $\mathsf{Gen}(\mathsf{PRG}(x))$ generates a PKE public key, using random coins derived from $\mathsf{PRG}(x)$ for PKE key generation,

- $\mathsf{Gen}(x)$ generates a PKE secret key, using random coins derived from $\mathsf{PRG}(x)$.

Note that $\mathsf{Gen}(x)$ outputs the matching PKE secret key to the public key output by $\mathsf{Gen}(\mathsf{PRG}(x))$. Furthermore, we use $\lambda + k$ as a security parameter for the indistinguishability obfuscation, and k for the PKE key generation. (Hence, with larger k, the keys produced by Gen become more secure.)

We note that the long-term security of our scheme relies *only* on the security of PRG. Moreover, the short-term security (which relies on the obfuscator and PKE) can be increased (by increasing k and replacing the CRS) without changing the PKI. Furthermore, we show that releasing short-term secret keys for previous CRSs does not harm the security of the current instance of the scheme. (We remark that a similar setup and technique has been used by [7] for a different purpose, in the context of non-interactive key exchange.)

Reconfigurable encryption in pairing-friendly groups. We also present a reconfigurable encryption scheme in a cyclic group $G = \langle g \rangle$ that admits a symmetric pairing $e : G \times G \to G_T$ into some target group $G_T = \langle g_T \rangle$. Both groups are of prime order $p > 2^\lambda$. The long-term assumption is the hardness of computing discrete logarithms in G, while the short-term assumption is the k-SCasc assumption from [11] over G (with a pairing).[1] To explain our scheme in a bit more detail, we adopt the notation of [11] and write $[x] \in G$ (resp. $[x]_T \in G_T$) for the group element g^x (resp. g_T^x), and similarly for vectors $[\vec{u}]$ and matrices $[\mathbf{A}]$ of group elements.

A long-term secret key is an exponent x, and the corresponding long-term public key is $[x]$. A CRS for a certain value $k \in \mathbb{N}$ is a uniform vector $[\vec{y}] \in G^k$ of group elements. The induced short-term public key is a matrix $[\mathbf{A}_x] \in G^{(k+1)\times k}$ derived from $[x]$, and the short-term secret key is a vector $[\vec{r}] \in G^{k+1}$ satisfying $\vec{r}^\top \cdot \mathbf{A}_x = \vec{y}$. An encryption of a message $m \in G_T$ is of the form

$$c \;=\; (\,[\mathbf{A}_x \cdot \vec{s}],\, [\vec{y}^\top \cdot \vec{s}]_T \cdot m\,)$$

for a uniformly chosen $[\vec{s}] \in G^k$. Intuitively, the k-SCasc assumption states that $[\mathbf{A}_x \cdot \vec{s}]$ is computationally indistinguishable from a random vector of group elements. This enables a security proof very similar to that for (dual) Regev encryption [18, 13] (see also [8]).

Hence, the long-term security of the above scheme is based on the discrete logarithm problem. Its short-term security relies on the k-SCasc assumption, where k can be adapted at runtime, without changing keys in the underlying PKI. Furthermore, we show that revealing previous short-term keys $[\vec{r}]$ does not harm the security of the current instance.[2]

We remark that [11] also present a less complex generalization of ElGamal to the k-SCasc assumption. Although they do not emphasize this property, their scheme allows to dynamically choose k at encryption time. However, their scheme does not in any obvious way allow to derive a short-term secret key that would be restricted to a given value of k. In other words, after, e.g., a key leakage, their scheme becomes insecure for all k, without the possibility of a reconfiguration.

Our reconfigurable signature scheme. We also construct a reconfigurable signature scheme in pairing-friendly groups. Its long-term security is based on the Computational Diffie-Hellman (CDH) assumption, and its short-term security can be based on any matrix assumption (e.g., on k-SCasc). Of course, efficient (non-reconfigurable) signature schemes from the CDH assumption already exist (e.g.,

[1] The k-SCasc assumption states that it is hard to distinguish vectors of group elements from a certain linear subspace from vectors of independently uniform group elements. Here, the parameter k determines the size of vectors, and – similar to the k-Linear assumption –, it is known that the k-SCasc assumption implies the $(k+1)$-SCasc assumption. In the generic group model, the $(k+1)$-SCasc assumption is also *strictly* weaker than the k-SCasc assumption [11]. Hence, increasing k leads to (at least generically) weaker assumptions.

[2] Currently, the best way to solve most problems in cyclic groups (such as k-SCasc or k-Linear instances) appears to be to compute discrete logarithms. In that sense, it would seem that the long-term and short-term security of our scheme are in a practical sense equivalent. Still, we believe that it is useful to offer solutions that give progressively stronger *provable* security guarantees (such as in our case with the k-SCasc assumption), if only to have fallback solutions in case of algorithmic advances, say, concerning the Decisional Diffie-Hellman problem.

Waters' signature scheme [23]). Compared to such schemes, our scheme still offers reconfigurability in case, e.g., short-term secret keys are leaked.

Roadmap. We start with some preliminaries in Section 2, followed by the definition of a reconfigurable encryption scheme and the security experiment in Section 3. In Section 4, we give the details of our two constructions for reconfigurable encryption. Finally, we treat reconfigurable signature schemes in Section 5.

2 Preliminaries

Notation. Throughout the paper, $\lambda, k, \ell \in \mathbb{N}$ denote security parameters. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For a probabilistic algorithm $\mathcal{A}$, we denote with $\mathcal{R}_{\mathcal{A}}$ the space of $\mathcal{A}$'s random coins. $y \leftarrow \mathcal{A}(x;r)$ denotes the process of running $\mathcal{A}$ on input x and with uniform randomness $r \in \mathcal{R}_{\mathcal{A}}$, and assigning y the result. We write $y \leftarrow \mathcal{A}(x)$ for $y \leftarrow \mathcal{A}(x;r)$ with uniform r. If $\mathcal{A}$'s running time, denoted by $\mathbf{T}(\mathcal{A})$, is polynomial in λ, then $\mathcal{A}$ is called probabilistic polynomial-time (PPT). We call a function η negligible if for every polynomial p there exists λ_0 such that for all $\lambda \geq \lambda_0$ holds $|\eta(\lambda)| \leq \frac{1}{p(\lambda)}$.

Concrete security. To formalize security of reconfigurable encryption schemes, we make use of the concept of concrete security as introduced in [3, 2]. Here one considers an explicit function for the adversarial advantage in breaking an assumption, a primitive, a protocol, etc. which is parameterized in the adversarial resources. More precisely, as usual let $\mathsf{Adv}^{\mathsf{x}}_{\mathcal{P},\mathcal{A}}(\lambda)$ denote the advantage function of an adversary $\mathcal{A}$ in winning some security experiment $\mathsf{Exp}^{\mathsf{x}}_{\mathcal{P},\mathcal{A}}(\lambda)$ defined for some cryptographic object $\mathcal{P}$ (e.g., a PKE scheme, the DDH problem, etc.) in the security parameter λ. For an integer $t \in \mathbb{N}$, we define the *concrete advantage* $\mathsf{CAdv}^{\mathsf{x}}_{\mathcal{P}}(t,\lambda)$ of breaking $\mathcal{P}$ with runtime t by

$$\mathsf{CAdv}^{\mathsf{x}}_{\mathcal{P}}(t,\lambda) := \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{x}}_{\mathcal{P},\mathcal{A}}(\lambda)\}, \tag{1}$$

where the maximum is over all $\mathcal{A}$ with time complexity t. It is straightforward to extend this definition to cryptographic objects defined in two security parameters which we introduce in this paper. In the following, if we are given an advantage function $\mathsf{Adv}^{\mathsf{x}}_{\mathcal{P},\mathcal{A}}(\lambda)$ for a cryptographic primitive $\mathcal{P}$ that we consider, the definition of the concrete advantage can then be derived as in (1). Asymptotic security (against non-uniform adversaries and when only one security parameter is considered) then means that $\mathsf{CAdv}^{\mathsf{x}}_{\mathcal{P}}(t(\lambda),\lambda)$ is negligible for all polynomials t in λ. Hence, if we only give the usual security definition for a cryptographic building block in the following its concrete security is also defined implicitly as described above.

Implicit representation. Let G be a cyclic group of order p generated by g. Then by $[a] := g^a$ we denote the *implicit representation* of $a \in \mathbb{Z}_p$ in G. To distinguish between implicit representations in two groups G and G_T, we use $[\cdot]$ and $[\cdot]_T$, respectively. The notation naturally extends to vectors and matrices of group elements.

Matrix-vector products. Sometimes, we will need to perform simple operations from linear algebra "in the exponent", aided by a pairing operation as necessary. Concretely, we will use the following operations: If a matrix $[\mathbf{A}] = [(a_{i,j})_{i,j}] \in G^{m\times n}$ is known "in the exponent", and a vector $\vec{u} = (u_i)_i \in \mathbb{Z}_p^n$ is known "in plain", then the product $[\mathbf{A}\cdot\vec{u}] \in G^m$ can be efficiently computed as $[(v_i)_i]$ for $[v_i] = \sum_{j=1}^n u_j \cdot [a_{i,j}]$. Similarly, inner products $[\vec{u}^\top \cdot \vec{v}]$ can be computed from $[\vec{u}]$ and $\vec{v}$ (or from $\vec{u}$ and $[\vec{v}]$). Finally, if only $[\mathbf{A}]$ and $[\vec{u}]$ are known (i.e., only "in the exponent"), still $[\mathbf{A}\cdot\vec{u}]_T$ can be computed in the target group, as $[(v_i)_i]_T$ for $[v_i]_T = \sum_{j=1}^n e([a_{i,j}],[u_j])$.

Symmetric pairing-friendly group generator. A symmetric pairing-friendly group generator is a probabilistic polynomial time algorithm $\mathcal{G}$ that takes as input a security parameter 1^λ and outputs a tuple $\mathbb{G} := (p, G, g, G_T, e)$ where

- G and G_T are cyclic groups of prime order p, $\lceil \log_2(p) \rceil = \lambda$ and $\langle g \rangle = G$
- $e : G \times G \longrightarrow G_T$ is an efficiently computable non-degenerate bilinear map

The Matrix Diffie-Hellman assumption ([11]). Let $k, q \in \mathbb{N}$ and $\mathcal{D}_k$ be an efficiently samplable matrix distribution over $\mathbb{Z}_q^{(k+1)\times k}$. The $\mathcal{D}_k$-Diffie-Hellman assumption ($\mathcal{D}_k$-MDDH) relative to a pairing-friendly group generator $\mathcal{G}$ states that for all PPT adversaries $\mathcal{A}$ it holds that

$$\mathsf{Adv}^{\mathcal{D}_k\text{-MDDH}}_{\mathcal{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathbb{G}, [\mathbf{A}, \mathbf{A}\vec{w}]) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{A}, \vec{u}]) = 1]|$$

is negligible in λ, where the probability is over the random choices $\mathbf{A} \leftarrow \mathcal{D}_k, \vec{w} \leftarrow \mathbb{Z}_q^k$ and $\vec{u} \leftarrow \mathbb{Z}_q^{k+1}$, $\mathbb{G} := (p, G, g, G_T, e) \leftarrow \mathcal{G}$ and the random coins of $\mathcal{A}$. Examples of $\mathcal{D}_k$-MDDH assumptions are the k-Lin assumption and the compact symmetric k-cascade assumption (k-SCasc or $\mathcal{SC}_k$-MDDH). For the latter the matrix distribution $\mathcal{SC}_k$ samples matrices of the form

$$\mathbf{A}_x := \begin{pmatrix} x & 0 & \dots & 0 & 0 \\ 1 & x & \dots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & x \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{Z}_n^{(k+1)\times k} \tag{2}$$

for uniformly random $x \leftarrow \mathbb{Z}_n$. In Section 4.2, we will consider a version of the SCasc assumption defined in two security parameters, λ (which defines the group order) and k.

PKE schemes. A public-key encryption (PKE) scheme PKE with message space $\mathcal{M}$ consists of three PPT algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$. Key generation $\mathsf{Gen}(1^\ell)$ outputs a public key pk and a secret key sk. Encryption $\mathsf{Enc}(pk, m)$ takes pk and a message $m \in \mathcal{M}$, and outputs a ciphertext c. Decryption $\mathsf{Dec}(sk, c)$ takes sk and a ciphertext c, and outputs a message m. For correctness, we want $\mathsf{Dec}(sk, c) = m$ for all $m \in \mathcal{M}$, all $(pk, sk) \leftarrow \mathsf{Gen}(1^\ell)$, and all $c \leftarrow \mathsf{Enc}(pk, m)$.

IND-CPA and IND-CCA security. Let PKE be a PKE scheme as above. For an adversary $\mathcal{A}$, consider the following experiment: first, the experiment samples $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ and runs $\mathcal{A}$ on input pk. Once $\mathcal{A}$ outputs two messages m_0, m_1, the experiment flips a coin $b \leftarrow \{0, 1\}$ and runs $\mathcal{A}$ on input $c^* \leftarrow \mathsf{Enc}(pk, m_b)$. We say that $\mathcal{A}$ wins the experiment iff $b' = b$ for $\mathcal{A}$'s final output b'. We denote $\mathcal{A}$'s advantage with $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(k) := |\Pr[\mathcal{A} \text{ wins}] - 1/2|$ and say that PKE is IND-CPA secure iff $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}}(k)$ is negligible for all PPT $\mathcal{A}$. Similarly, write $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(k) := |\Pr[\mathcal{A} \text{ wins}] - 1/2|$ for $\mathcal{A}$'s winning probability when $\mathcal{A}$ additionally gets access to a decryption oracle $\mathsf{Dec}(sk, \cdot)$ at all times. (To avoid trivialities, $\mathcal{A}$ may not query Dec on c^*, though.)

PRGs. Informally, a pseudorandom generator (PRG) is a deterministic algorithm that maps a short random bit string (called seed) to a longer pseudo-random bitstring. More formally, let $p(\cdot)$ be a polynomial such that $p(\lambda) > \lambda$ for all $\lambda \in \mathbb{N}$ and let PRG be a deterministic polynomial-time algorithm which on input of a bit string in $\{0,1\}^\lambda$ returns a bit string in $\{0,1\}^{p(\lambda)}$ (also denoted by $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{p(\lambda)}$). The security of PRG is defined through the advantage

$$\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{PRG},D}(\lambda) := |\Pr[1 \leftarrow D(\mathsf{PRG}(x))] - \Pr[1 \leftarrow D(r)]|\,,$$

where D is a distinguisher, $x \leftarrow \{0,1\}^\lambda$ and $r \leftarrow \{0,1\}^{p(\lambda)}$.

Indistinguishability Obfuscation ($i\mathcal{O}$). For our construction in Section 4.1, we make use of indistinguishability obfuscators for polynomial-size circuits. Intuitively, such an algorithm is able to obfuscate two equivalent circuits in a way such that a PPT adversary who receives the two obfuscated circuits as input is not able to distinguish them. The following definition is taken from [12].

Definition 2.1 (Indistinguishability Obfuscator)**.** *A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\ell\}$ if the following conditions are satisfied:*

- *For all security parameters $\ell \in \mathbb{N}$, for all $C \in \mathcal{C}_\ell$, for all inputs x, we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\ell, C)] = 1$$

- *For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\ell \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\ell$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then*

$$\mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},D}(\ell) := |\Pr[1 \leftarrow D(i\mathcal{O}(\ell, C_0))] - \Pr[1 \leftarrow D(i\mathcal{O}(\ell, C_1))]| \leq \alpha(\ell)$$

Note that an $i\mathcal{O}$ candidate for circuit classes $\{\mathcal{C}_\ell\}$, where the input size as well as the maximum circuit size are polynomials in ℓ has been proposed in [12].

Puncturable PRF. Informally speaking, a puncturable (or constrained) PRF $F_K : \{0,1\}^{n(\ell)} \to \{0,1\}^{p(\ell)}$ is a PRF for which it is possible to constrain the key K (i.e., derive a new key K_S) in order to exclude a certain subset $S \subset \{0,1\}^{n(\ell)}$ of the domain of the PRF. (Note that this means that $F_{K_S}(x)$ is not defined for $x \in S$ and equal to $F_K(x)$ for $x \notin S$.) Given the punctured key K_S, an adversary may not be able to distinguish $F_K(x)$ from a random $y \in \{0,1\}^{p(\ell)}$ for $x \in S$. The following definition adapted from [19] formalizes this notion.

Definition 2.2. *A puncturable family of PRFs F is given by three PPT algorithms Gen_F, $\mathsf{Puncture}_F$, and Eval_F, and a pair of computable functions $(n(\cdot), p(\cdot))$, satisfying the following conditions:*

- *For every $S \subset \{0,1\}^{n(\ell)}$, for all $x \in \{0,1\}^{n(\ell)}$ where $x \notin S$, we have that:*

$$\Pr[\mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_S, x) : K \leftarrow \mathsf{Gen}_F(1^\ell), K_S \leftarrow \mathsf{Puncture}_F(K, S)] = 1$$

- *For every PPT adversary $\mathcal{A}$ such that $\mathcal{A}(1^\ell)$ outputs a set $S \subset \{0,1\}^{n(\ell)}$ and a state state, consider an experiment where $K \leftarrow \mathsf{Gen}_F(1^\ell)$ and $K_S = \mathsf{Puncture}_F(K, S)$. Then the advantage $\mathsf{Adv}^{\mathsf{pprf}}_{F,\mathcal{A}}(\ell)$ of $\mathcal{A}$ defined by*

$$\left|\Pr[1 \leftarrow \mathcal{A}(\mathsf{state}, K_S, \mathsf{Eval}_F(K, S))] - \Pr[1 \leftarrow \mathcal{A}(\mathsf{state}, K_S, U_{p(\ell)\cdot|S|})]\right|$$

  is negligible, where $\mathsf{Eval}_F(K, S)$ denotes the concatenation of $\mathsf{Eval}_F(K, x_i), i = 1, ..., m$, where $S = \{x_1, \ldots, x_m\}$ is the enumeration of the elements in S in lexicographic order, and U_t denotes the uniform distribution over t bits.

To simplify notation, we write $F_K(x)$ instead of $\mathsf{Eval}_F(K, x)$. Note that if one-way functions exist, then there also exist a puncturable PRF family for any efficiently computable functions $n(\ell)$ and $p(\ell)$.

3 Definitions

The idea behind our concept of a reconfigurable public key cryptosystem is very simple: instead of directly feeding a PKI into the algorithms of the cryptosystem, we add some precomputation routines to derive a temporary short-term PKI. This PKI is then used by the cryptosystem. Instructions on how to derive and when to update the short-term PKI are given by a trusted entity. Our concept is quite modular and, thus, is applicable to other cryptosystems as well. In this section, we consider the case of reconfigurable encryption.

In Definition 3.1, we give a formal description of a reconfigurable public key encryption (RPKE) scheme. An RPKE scheme is a multi-user system which is setup (once) by some trusted entity generating public system parameters given a long-term security parameter 1^λ. Based on these public parameters, each user generates his long-term key pair. Moreover, the entity uses the public parameters to generate a common reference string defining a certain (short-term) security level k. Note

that only this CRS is being updated when a new short-term security level for the system should be established. The current CRS is distributed to all users, who derive their short-term secret and public keys for the corresponding security level from their long-term secret and public keys and the CRS. Encryption and decryption of messages works as in a standard PKE using the short-term key pair of a user.

Definition 3.1. *A reconfigurable public-key encryption (RPKE) scheme* RPKE *consists of the following PPT algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *receives a long-term security parameter* 1^λ *as input, and returns (global) long-term public parameters* $\mathcal{PP}$.
- $\mathsf{MKGen}(\mathcal{PP})$ *takes the long-term public parameters* $\mathcal{PP}$ *as input and returns the long-term public and private key* (mpk, msk) *of a user.*
- $\mathsf{CRSGen}(\mathcal{PP}, 1^k)$ *is given the long-term public parameters* $\mathcal{PP}$*, a short-term security parameter* 1^k*, and returns a (global) short-term common reference string* CRS*. We assume that the message space* $\mathcal{M}$ *is defined as part of* CRS.
- $\mathsf{PKGen}(CRS, mpk)$ *takes the CRS* CRS *as well as the long-term public key* mpk *of a user as input and returns a short-term public key* pk *for this user.*
- $\mathsf{SKGen}(CRS, msk)$ *takes the CRS* CRS *as well as the long-term secret key* msk *of a user as input and returns a short-term secret key* sk *for this user.*
- $\mathsf{Enc}(pk, m)$ *receives a user's short-term public key* pk *and a message* $m \in \mathcal{M}$ *as input and returns a ciphertext* c.
- $\mathsf{Dec}(sk, c)$ *receives a user's short-term secret key* sk *and a ciphertext* c *as input and returns* $m \in \mathcal{M} \cup \{\perp\}$.

We call RPKE *correct if for all* $\lambda, k \in \mathbb{N}$, $\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$, $(mpk, msk) \leftarrow \mathsf{MKGen}(\mathcal{PP})$, $CRS \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$, $m \in \mathcal{M}$, $pk \leftarrow \mathsf{PKGen}(CRS, mpk)$, $sk \leftarrow \mathsf{SKGen}(CRS, msk)$, *and all* $c \leftarrow \mathsf{Enc}(pk, m)$, *it holds that* $\mathsf{Dec}(sk, c) = m$.

Security. Our security experiment for RPKE systems given in Figure 1 is inspired by the notion of IND-CCA (IND-CPA) security, extended to the more involved key generation phase of a reconfigurable encryption scheme. Note that we provide the adversary with a secret key oracle for deprecated short-term keys. The intuition behind our security definition is that we can split the advantage of an adversary into three parts. One part (called f_1 in Definition 3.2) reflects its advantage in attacking the subsystem of an RPKE that is only responsible for long-term security (λ). Another part (f_2) represents its advantage in attacking the subsystem that is only responsible for short-term security (k). The remaining part (f_3) stands for its advantage in attacking the subsystem that links the long-term with the short-term security subsystem (e.g., short-term key derivation). We demand that all these advantages are negligible in the corresponding security parameter, i.e., part one in λ, part two in k, and part three in both λ (where k is fixed) and in k (where λ is fixed).

Note that it is not reasonable to simply demand that the overall advantage is negligible in λ and in k. For instance, consider the advantage function $\mathsf{CAdv}(t(\lambda, k), \lambda, k) \leq 2^{-\lambda} + 2^{-k} + 2^{-(\lambda+k)}$. Intuitively, we would like to call an RPKE exhibiting this bound as secure. Unfortunately, it is neither negligible in λ nor in k.

Definition 3.2. *Let* RPKE *be an RPKE scheme according to Definition 3.1. Then we define the advantage of an adversary* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE},\mathcal{A}}(\lambda, k) := \left| \Pr[\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE},\mathcal{A}}(\lambda, k) = 1] - \frac{1}{2} \right|$$

Experiment $\mathsf{Exp}^{\text{r-ind-cca}}_{\mathsf{RPKE},\mathcal{A}}(\lambda, k)$

$\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$

$(mpk, msk) \leftarrow \mathsf{MKGen}(\mathcal{PP})$

$state \leftarrow \mathcal{A}^{\mathsf{Break}(\mathcal{PP}, msk, \cdot)}(1^\lambda, 1^k, \mathcal{PP}, mpk, \text{“learn”})$

$CRS^* \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$

$sk^* \leftarrow \mathsf{SKGen}(CRS^*, msk)$

$pk^* \leftarrow \mathsf{PKGen}(CRS^*, mpk)$

$(m_0, m_1, state') \leftarrow \mathcal{A}^{\mathsf{Dec}(sk^*, \cdot)}(CRS^*, state, \text{“select”})$

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

$c^* \leftarrow \mathsf{Enc}(pk^*, m_b)$

$out_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathsf{Dec}(sk^*, \cdot)}(c^*, state', \text{“guess”})$

Let $k_1, \ldots, k_\ell$ be the inputs sent to the Break-Oracle by $\mathcal{A}$. On input k_i, the Break-Oracle returns $CRS_{k_i} \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^{k_i})$ as well as $sk_{k_i} \leftarrow \mathsf{SKGen}(CRS_{k_i}, msk)$ to $\mathcal{A}$.

Return 1 if $k_i < k$ for all i, $|m_0| = |m_1|$, $out_{\mathcal{A}} = b$, and c^* has never been sent as input to the Dec-Oracle. Otherwise, return 0.

Figure 1: R-IND-CCA experiment for reconfigurable PKE.

where $\mathsf{Exp}^{\text{r-ind-cca}}_{\mathsf{RPKE},\mathcal{A}}$ *is the experiment given in Figure 1. The concrete advantage* $\mathsf{CAdv}^{\text{r-ind-cca}}_{\mathsf{RPKE}}(t, \lambda, k)$ *of adversaries against* RPKE *with time complexity* t *follows canonically (cf. Section 2).*

An RPKE scheme RPKE *is then called R-IND-CCA secure if for all polynomials* $t(\lambda, k)$*, there exist positive functions* $f_1 : \mathbb{N}^2 \to \mathbb{R}_0^+$*,* $f_2 : \mathbb{N}^2 \to \mathbb{R}_0^+$*, and* $f_3 : \mathbb{N}^3 \to \mathbb{R}_0^+$ *as well as polynomials* $t_1(\lambda, k)$*,* $t_2(\lambda, k)$*, and* $t_3(\lambda, k)$ *such that*

$$\mathsf{CAdv}^{\text{r-ind-cca}}_{\mathsf{RPKE}}(t(\lambda, k), \lambda, k) \leq f_1(t_1(\lambda, k), \lambda) + f_2(t_2(\lambda, k), k) + f_3(t_3(\lambda, k), \lambda, k)$$

for all λ, k*, and the following conditions are satisfied for* f_1, f_2, f_3*:*

- *For all* $k \in \mathbb{N}$ *it holds that* $f_1(t_1(\lambda, k), \lambda)$ *is negligible in* λ
- *For all* $\lambda \in \mathbb{N}$ *it holds that* $f_2(t_2(\lambda, k), k)$ *is negligible in* k
- *For all* $k \in \mathbb{N}$ *it holds that* $f_3(t_3(\lambda, k), \lambda, k)$ *is negligible in* λ
- *For all* $\lambda \in \mathbb{N}$ *it holds that* $f_3(t_3(\lambda, k), \lambda, k)$ *is negligible in* k

We define R-IND-CPA security analogously with respect to the modified experiment $\mathsf{Exp}^{\text{r-ind-cpa}}_{\mathsf{RPKE},\mathcal{A}}(\lambda, k)$*, which is identical to* $\mathsf{Exp}^{\text{r-ind-cca}}_{\mathsf{RPKE},\mathcal{A}}(\lambda, k)$ *except that* $\mathcal{A}$ *has no access to an* Dec-*Oracle.*

In Section 1 we already sketched an IBE-based RPKE scheme that would be secure in the sense of Definition 3.2. However, for this RPKE it is obvious that f_2 and f_3 can be set to be the zero function, meaning that the adversarial advantage cannot be decreased by increasing k. In this paper we are not interested in such schemes.

Of course, one can think of several reasonable modifications to the security definition given above. For instance, one may want to omit the “learn” stage in the experiment and instead give the algorithm access to the Break-Oracle during the “select” and “guess” stages. Fortunately, it turned out that most of these reasonable, slight modifications lead to a definition which is equivalent to the simple version we chose.

4 Constructions

4.1 Reconfigurable Encryption from Indistinguishability Obfuscation

We can build a R-IND-CCA (R-IND-CPA) secure reconfigurable encryption scheme from any IND-CCA (IND-CPA) secure PKE using indistinguishable obfuscation and puncturable PRFs. The basic idea is simple: We obfuscate a circuit which on input of the long-term public or secret key, where the public key is simply the output of a PRG on input of the secret key, calls the key generator of the PKE scheme using random coins derived by means of the PRF. It outputs the public key of the PKE scheme if the input to the circuit was the long-term public key and the secret key if the input was the long-term secret key.

Ingredients. Let $\mathsf{PKE}_{\mathsf{CCA}} = (\mathsf{Gen}_{\mathsf{CCA}}, \mathsf{Enc}_{\mathsf{CCA}}, \mathsf{Dec}_{\mathsf{CCA}})$ be an IND-CCA secure encryption scheme. Assuming the first component of the key pair that $\mathsf{Gen}_{\mathsf{CCA}}(1^\ell)$ outputs is the public key, we define the PPT algorithms $\mathsf{PKGen}_{\mathsf{CCA}}(1^\ell) := \#1(\mathsf{Gen}_{\mathsf{CCA}}(1^\ell))$ and $\mathsf{SKGen}_{\mathsf{CCA}}(1^k) := \#2(\mathsf{Gen}_{\mathsf{CCA}}(1^k))$ which run $\mathsf{Gen}_{\mathsf{CCA}}(1^\ell)$ and output only the public key or only the secret key, respectively. By writing $\mathsf{Gen}_{\mathsf{CCA}}(1^k; r)$, $\mathsf{PKGen}_{\mathsf{CCA}}(1^k; r)$, $\mathsf{SKGen}_{\mathsf{CCA}}(1^k; r)$ we will denote the act of fixing the randomness used by $\mathsf{Gen}_{\mathsf{CCA}}$ for key generation to be r, a random bit string of sufficient length. For instance, r could be of polynomial length $p(k)$, where p equals the runtime complexity of $\mathsf{Gen}_{\mathsf{CCA}}$. We allow r to be longer than needed and assume that any additional bits are simply ignored by $\mathsf{Gen}_{\mathsf{CCA}}$.[3] Furthermore, let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a pseudo-random generator and F be a family of puncturable PRFs mapping $n(\ell) := 2\ell$ bits to $p(\ell)$ bits. For $i \in \mathbb{N}$ we define $\mathsf{pad}_i : \{0,1\}^* \to \{0,1\}^*$ as the function which appends i zeroes to a given bit string. As a last ingredient, we need an indistinguishability obfuscator $i\mathcal{O}(\ell, C)$ for a class of circuits of size at most $q(\ell)$, where q is a suitable polynomial in $\ell = \lambda + k$ which upper bounds the size of the circuit $\mathsf{Gen}(a, b)$ to be defined as part of CRSGen.[4]

Our scheme. With the ingredients described above our RPKE $\mathsf{RPKE}_{i\mathcal{O}}$ can be defined as in Figure 2. Note that the security parameter ℓ used in the components for deriving short-term keys from long-term keys, i.e., F and $i\mathcal{O}$, is set to $\lambda + k$. That means, it increases (and the adversarial advantage becomes negligible) with both, the long-term and the short-term security parameter. (Alternative choices with the same effect like $\ell = \frac{\lambda}{2} + k$ are also possible.) Since the components which generate and use the short-term secrets depend on k, the security of the scheme can be increased by raising k. As a somewhat disturbing side-effect of our choice of ℓ, the domain of F, which is used to map the long-term public key $mpk \in \{0,1\}^{2\lambda}$ to a pseudo-random string to be used by $\mathsf{Gen}_{\mathsf{CCA}}$, is actually too large. Hence, we have to embed 2λ-bit strings into $2(\lambda + k)$-bit strings by applying pad_{2k}.

Security. R-IND-CCA security of $\mathsf{RPKE}_{i\mathcal{O}}$ follows from the following Lemma.

Lemma 4.1. *Let $t \in \mathbb{N}$ be given and let t' denote the maximal runtime of the experiment $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\cdot}(\lambda, k)$ involving arbitrary adversaries with runtime t. Then it holds that*

$$\begin{aligned} \mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}}}(t, \lambda, k) &\leq \tfrac{1}{2^\lambda} + \mathsf{CAdv}^{\mathsf{prg}}_{\mathsf{PRG}}(s_1, \lambda) + \mathsf{CAdv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}}}(s_2, k) \\ &+ \mathsf{CAdv}^{\mathsf{pprf}}_{\mathsf{F}}(s_3, \lambda + k) + \mathsf{CAdv}^{\mathsf{io}}_{i\mathcal{O}}(s_4, \lambda + k) \end{aligned} \tag{3}$$

where $t' \approx s_1 \approx s_2 \approx s_3 \approx s_4$.

Proof. The following reduction will be in the non-uniform adversary setting. Consider an adversary $\mathcal{A}$ against $\mathsf{RPKE}_{i\mathcal{O}}$ for fixed security parameters λ and k who has an advantage denoted by $\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)$. We will first show that $\mathcal{A}$ can be turned into adversaries

[3] Equivalently, we could apply a truncate function $\mathsf{trunc}_{p(k)} : \{0,1\}^* \to \{0,1\}^{p(k)}$ which outputs the $p(k)$ most significant bits of a given input.

[4] Note that actually q must be chosen as an upper bound of both Gen and Gen', where the latter is defined in the security proof.

$\underline{\mathsf{Setup}(1^\lambda)}$
return $\mathcal{PP} := (1^\lambda)$

$\underline{\mathsf{MKGen}(\mathcal{PP})}$
$x \leftarrow \{0,1\}^\lambda$
$mpk := \mathsf{PRG}(x)$
$msk := x$
return (mpk, msk)

$\underline{\mathsf{CRSGen}(\mathcal{PP}, 1^k)}$
$K \leftarrow \mathsf{Gen}_\mathsf{F}(1^{\lambda+k})$

$$\mathsf{Gen}(a,b) := \begin{cases} \mathsf{PKGen}_{\mathsf{CCA}}(1^k; \mathsf{F}_K(\mathsf{pad}_{2k}(a))), & b = 0 \wedge a \in \{0,1\}^{2\lambda} \\ \mathsf{SKGen}_{\mathsf{CCA}}(1^k; \mathsf{F}_K(\mathsf{pad}_{2k}(\mathsf{PRG}(a)))), & b = 1 \wedge a \in \{0,1\}^{\lambda} \\ \bot, & \text{else} \end{cases}$$

$\mathsf{iOGen} \leftarrow i\mathcal{O}(\lambda + k, \mathsf{Gen}(a,b))$
return $CRS := (\mathsf{iOGen})$

$\underline{\mathsf{PKGen}(CRS, mpk)}$
parse $\mathsf{iOGen} := CRS$
return $\mathsf{iOGen}(mpk, 0)$

$\underline{\mathsf{SKGen}(CRS, msk)}$
parse $\mathsf{iOGen} := CRS$
return $\mathsf{iOGen}(msk, 1)$

$\underline{\mathsf{Enc}(pk, m)}$
return $\mathsf{Enc}_{\mathsf{CCA}}(pk, m)$

$\underline{\mathsf{Dec}(sk, c)}$
return $\mathsf{Dec}_{\mathsf{CCA}}(sk, c)$

Figure 2: Our $i\mathcal{O}$-based RPKE scheme $\mathsf{RPKE}_{i\mathcal{O}}$

- $\mathcal{B}$ against PRG for fixed security parameter λ with advantage $\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{PRG},\mathcal{B}_k}(\lambda)$,
- $\mathcal{C}$ against $i\mathcal{O}$ for fixed security parameter $\lambda + k$ with advantage $\mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},\mathcal{C}}(\lambda + k)$,
- $\mathcal{D}$ against F for fixed security parameter $\lambda + k$ with advantage $\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{F},\mathcal{D}}(\lambda + k)$, and
- $\mathcal{E}$ against $\mathsf{PKE}_{\mathsf{CCA}}$ for fixed security parameter k with advantage $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}},\mathcal{E}}(k)$

such that the advantage $\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)$ is upper bounded by

$$\frac{1}{2^\lambda} + \mathsf{Adv}^{\mathsf{prg}}_{\mathsf{PRG},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}},\mathcal{E}}(k) + \mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},\mathcal{C}}(\lambda + k) + \mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{F},\mathcal{D}}(\lambda + k). \tag{4}$$

After that, we will argue that from Equation 4 the upper bound on the concrete advantage stated in Equation 3 from our Lemma follows.

Throughout the reduction proof, let $\mathsf{Adv}^{\mathrm{Game}_i}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)$ denote the advantage of $\mathcal{A}$ in winning Game i for fixed λ, k.

Game 1 is the real experiment $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}$. So we have

$$\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) = \mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k). \tag{5}$$

Game 2 is identical to Game 1 except that a short-term secret key returned by the Break-Oracle on input $k' < k$ is computed by executing

$$\mathsf{SKGen}_{\mathsf{CCA}}(1^{k'}; \mathsf{F}_K(\mathsf{pad}_{2k'}(mpk)))$$

instead of calling $\mathsf{SKGen}(CRS_{k'}, msk)$, where $CRS_{k'} \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^{k'})$ and $K \leftarrow \mathsf{Gen}_\mathsf{F}(1^{\lambda+k'})$ is the corresponding PRF key generated in the scope of $\mathsf{CRSGen}(\mathcal{PP}, 1^{k'})$. Similarly, the challenge secret

key sk^* is computed by the challenger by executing

$$\mathsf{SKGen}_{\mathsf{CCA}}(1^k; \mathsf{F}_{K^*}(\mathsf{pad}_{2k}(mpk))),$$

and not by calling $\mathsf{SKGen}(CRS^*, msk)$, where CRS^* denotes the challenge CRS and K^* the PRF key used in the process of generating CRS^* by applying $\mathsf{CRSGen}(\mathcal{PP}, 1^k)$. In this way, msk is not used in the game anymore after $mpk = \mathsf{PRG}(msk)$ has been generated. Obviously, this change cannot be noticed by $\mathcal{A}$ and so we have

$$\mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) = \mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k). \tag{6}$$

Game 3 is identical to Game 2 except that the challenge long-term public key is no longer computed as $mpk = \mathsf{PRG}(msk)$ but set to be a random bit string $r \leftarrow \{0,1\}^{2\lambda}$. Note with the change introduced in Game 2, we achieved that this game only depended on $\mathsf{PRG}(msk)$ but not on msk itself. Hence, we can immediately build an adversary $\mathcal{B}$ against PRG for (fixed) security parameter λ from a distinguisher between Games 1 and 2 with advantage

$$\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{PRG},\mathcal{B}_k}(\lambda) = \left|\mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) - \mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)\right|. \tag{7}$$

As a consequence, in Game 3 nothing at all is leaked about msk.

The PRG adversary $\mathcal{B}$ receives a bit string $y \in \{0,1\}^{2\lambda}$ from the PRG challenger which is either random (as in Game 3) or the output of $\mathsf{PRG}(x)$ for $x \leftarrow \{0,1\}^{\lambda}$ (as in Game 2). It computes $\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$, $CRS^* \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$, and sets $mpk := y$. Note that due to the changes in Game 2 the key msk (which would be the unknown x) is not needed to execute the experiment. Then it runs $\mathcal{A}$ on input $\mathcal{PP}$ and mpk. A Break-Query is handled as described in Game 2, i.e., sk is computed by $\mathcal{B}$ based on mpk. The challenge short-term key sk^* is computed in the same way from mpk. In this way, $\mathcal{B}$ can perfectly simulate the Dec-Oracle when it runs $\mathcal{A}$ on input CRS^*. When receiving two messages m_0 and m_1 from the adversary, $\mathcal{B}$ returns $c^* \leftarrow \mathsf{Enc}(pk^*, m_b)$ for random b where pk^* has been generated as usual from mpk. Then $\mathcal{B}$ forwards the final output of $\mathcal{A}$. Clearly, if y was random $\mathcal{B}$ perfectly simulated Game 3, otherwise it simulated Game 2.

To introduce the changes in **Game** 4, let

$$K^*_{\{\mathsf{pad}_{2k}(mpk)\}} := \mathsf{Puncture}_{\mathsf{F}}(K^*, \{\mathsf{pad}_{2k}(mpk)\})$$

denote the key K^* (used in the construction of CRS^*) where we punctured out mpk (represented as an element of $\{0,1\}^{2(\lambda+k)}$). This implies that $\mathsf{F}_{K^*_{\{\mathsf{pad}_{2k}(mpk)\}}}(a)$ is undefined for $a = \mathsf{pad}_{2k}(mpk)$. Now, we set $r := \mathsf{F}_{K^*}(\mathsf{pad}_{2k}(mpk))$ and the challenge short-term keys $pk^* := \mathsf{PKGen}_{\mathsf{CCA}}(1^k; r)$ and $sk^* := \mathsf{SKGen}_{\mathsf{CCA}}(1^k; r)$. Those keys are computed in the experiment immediately after the generation of the long-term key pair (mpk, msk). This is equivalent to the way these keys have been computed in Game 3. Additionally, we replace $\mathsf{Gen}(a, b)$ in CRSGen for the challenge security level k by

$$\mathsf{Gen}'(a,b) := \begin{cases} pk^*, & b = 0 \wedge a = mpk \\ \mathsf{PKGen}_{\mathsf{CCA}}(1^k; \mathsf{F}_{K^*_{\{\mathsf{pad}_{2k}(mpk)\}}}(\mathsf{pad}_{2k}(a))), & b = 0 \wedge a \in \{0,1\}^{2\lambda} \setminus \{mpk\} \\ \mathsf{SKGen}_{\mathsf{CCA}}(1^k; \mathsf{F}_{K^*_{\{\mathsf{pad}_{2k}(mpk)\}}}(\mathsf{pad}_{2k}(\mathsf{PRG}(a)))), & b = 1 \wedge a \in \{0,1\}^{\lambda} \\ \bot, & \text{else} \end{cases}$$

CRS^* will now include the obfuscated circuit $\mathsf{iOGen}' \leftarrow i\mathcal{O}(\lambda + k, \mathsf{Gen}'(a, b))$.

We now verify that the circuits Gen and Gen' are indeed equivalent (most of the time). Obviously, it holds that $\mathsf{Gen}(a, 0) = \mathsf{Gen}'(a, 0)$ for all $a \in \{0,1\}^{2\lambda}$: the precomputed value pk^* results from running

$\mathsf{PKGen}_{\mathsf{CCA}}(1^{\lambda+k}; \mathsf{F}_{K^*}(\mathsf{pad}_{2k}(mpk)))$ which is exactly what $\mathsf{Gen}(mpk, 0)$ would run too. Moreover, we have

$$\mathsf{F}_{K^*}(\mathsf{pad}_{2k}(a)) = \mathsf{F}_{K^*_{\{\mathsf{pad}_{2k}(mpk)\}}}(\mathsf{pad}_{2k}(a))$$

for all $a \in \{0,1\}^{2\lambda} \setminus \{mpk\}$. Let us now consider $\mathsf{Gen}'(a, 1)$ for $a \in \{0,1\}^{\lambda}$. Remember that starting with Game 3, mpk is a random element from $\{0,1\}^{2\lambda}$. That means, with probability at least $1 - \frac{1}{2^{\lambda}}$ we have that mpk is not in the image of PRG and, thus,

$$\mathsf{F}_{K^*}(\mathsf{pad}_{2k}(\mathsf{PRG}(a))) = \mathsf{F}_{K^*_{\{\mathsf{pad}_{2k}(mpk)\}}}(\mathsf{pad}_{2k}(\mathsf{PRG}(a)))$$

for all $a \in \{0,1\}^{\lambda}$. Hence, with probability $1 - \frac{1}{2^{\lambda}}$ the circuits Gen and Gen' are equivalent for all inputs. So a distinguisher between Game 4 and Game 3 can be turned into an adversary $\mathcal{C}$ against $i\mathcal{O}$ for security parameter $\lambda + k$ with advantage

$$\mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},\mathcal{C}}(\lambda + k) \geq \left|\mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) - \mathsf{Adv}^{\mathrm{Game}_4}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)\right| - \frac{1}{2^{\lambda}}. \tag{8}$$

$\mathcal{C}$ computes $\mathcal{PP} \leftarrow \mathsf{Setup}(1^{\lambda})$ and $mpk \leftarrow \{0,1\}^{2\lambda}$. Then it chooses a PPRF $\mathsf{F} : \{0,1\}^{2(\lambda+k)} \to \{0,1\}^{p(\lambda+k)}$ and a corresponding key $K^* \leftarrow \mathsf{Gen}_{\mathsf{F}}(1^{\lambda+k})$. Using these ingredients it sets up circuits $C_0 := \mathsf{Gen}$ according to the definition from Game 3 and $C_1 := \mathsf{Gen}'$ according to the definition from Game 4. As explained above, with probability $1-\frac{1}{2^{\lambda}}$ these circuits are equivalent for all inputs. CRS^* is then set as the output of the $i\mathcal{O}$ challenger for security parameter $\lambda + k$ on input of the circuits C_0 and C_1.[5] sk^* and pk^* can either be computed as defined in Game 3 or as in Game 4. As both ways are equivalent, it does not matter for the reduction. The remaining parts of Game 3 and Game 4 are identical. In particular, Break-Queries of $\mathcal{A}$ can be handled without knowing msk. The output bit of the third and final execution of $\mathcal{A}$ is simply forwarded by $\mathcal{C}$ to the $i\mathcal{O}$ challenger.

Game 5 is identical to Game 4 except that the value r is chosen as a truly random string from $\{0,1\}^{p(\lambda+k)}$ and not set to $\mathsf{F}_{K^*}(\mathsf{pad}_{2k}(mpk))$. As besides r, Game 4 did not depend on K^* anymore but only on $K^*_{\{\mathsf{pad}_{2k}(mpk)\}}$, a distinguisher between Game 4 and Game 5 can directly be turned into an adversary $\mathcal{D}$ against the pseudorandomness of the puncturable PRF family for security parameter $\lambda + k$. Thus, we have

$$\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{F},\mathcal{D}}(\lambda + k) = \left|\mathsf{Adv}^{\mathrm{Game}_4}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) - \mathsf{Adv}^{\mathrm{Game}_5}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k)\right|. \tag{9}$$

$\mathcal{D}$ computes $\mathcal{PP} \leftarrow \mathsf{Setup}(1^{\lambda})$, $mpk \leftarrow \{0,1\}^{2\lambda}$, and chooses a PPRF $\mathsf{F} : \{0,1\}^{2(\lambda+k)} \to \{0,1\}^{p(\lambda+k)}$. Then it sends $\mathsf{pad}_{2k}(mpk)$ to its challenger who chooses a key $K^* \leftarrow \mathsf{Gen}_{\mathsf{F}}(1^{\lambda+k})$ and computes the punctured key $K^*_{\{\mathsf{pad}_{2k}(mpk)\}}$. Furthermore, the challenger sets $r_0 := \mathsf{F}_{K^*}(\mathsf{pad}_{2k}(mpk))$ and $r_1 \leftarrow \{0,1\}^{p(\lambda+k)}$. It chooses $b \leftarrow \{0,1\}$ and sends r_b along with $K^*_{\{\mathsf{pad}_{2k}(mpk)\}}$ to $\mathcal{D}$. $\mathcal{D}$ sets $r := t_b$, $pk^* := \mathsf{PKGen}_{\mathsf{CCA}}(1^k; r)$ and $sk^* := \mathsf{SKGen}_{\mathsf{CCA}}(1^k; r)$. Using the given punctured key $K^*_{\{\mathsf{pad}_{2k}(mpk)\}}$, $\mathcal{D}$ can also generate CRS^* as described in Game 4. The rest of the reduction is straightforward. The output bit of the final execution of $\mathcal{A}$ is simply forwarded by $\mathcal{C}$ to its challenger. If $b = 0$, $\mathcal{D}$ perfectly simulates Game 4, otherwise it simulates Game 5.

Now, observe that in Game 5, the keys pk^* and sk^* are generated using $\mathsf{Gen}_{\mathsf{CCA}}$ with a uniformly chosen random string r on its random tape. In particular, pk^* and sk^* are completely independent of the choice of mpk and msk. After the generation of these short-term keys, the adversary has access to the Break-Oracle, which, of course, will also not yield any additional information about them since the output of this oracle only depends on independent random choices like mpk and the PRF keys K. The remaining steps of Game 5 correspond to the regular IND-CCA game for $\mathsf{PKE}_{\mathsf{CCA}}$ except

[5] C_0 and C_1 are assumed to be of the same size, otherwise the smaller one is padded accordingly.

that the adversary is given the additional input CRS^*, which however only depends on pk^*, and the independent choices mpk and K^*. So except for pk^* (which is the output of $\mathsf{PKGen}(CRS^*, mpk)$), the adversary does not get any additional useful information from CRS^* (which he could not have computed by himself). Hence, it is easy to construct an IND-CCA adversary $\mathcal{E}$ against $\mathsf{PKE}_{\mathsf{CCA}}$ for security parameter k from $\mathcal{A}$ which has the same advantage as $\mathcal{A}$ in winning Game 5, i.e.,

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}},\mathcal{E}}(k) = \mathsf{Adv}^{\mathrm{Game}_5}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k). \tag{10}$$

$\mathcal{E}$ computes $\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ and $mpk \leftarrow \{0,1\}^{2\lambda}$. Break-Queries from $\mathcal{A}$ can be answered by $\mathcal{E}$ only based on mpk (as described in Game 2). Then $\mathcal{E}$ receives pk^* generated using $\mathsf{Gen}_{\mathsf{CCA}}(1^k)$ from the IND-CCA challenger. To compute CRS^*, $\mathcal{E}$ chooses a PPRF $\mathsf{F} : \{0,1\}^{2(\lambda+k)} \to \{0,1\}^{p(\lambda+k)}$, the corresponding key $K^* \leftarrow \mathsf{Gen}_{\mathsf{F}}(1^{\lambda+k})$ and sets the punctured key $K^*_{\{\mathsf{pad}_{2k}(mpk)\}}$. Using these ingredients, Gen' can be specified as in Game 4 and its obfuscation equals CRS^*. When $\mathcal{E}$ runs $\mathcal{A}$ on input CRS^*, $\mathcal{A}$'s queries to the Dec-Oracle are forwarded to the IND-CCA challenger. Similarly, the messages m_0 and m_1 that $\mathcal{A}$ outputs are sent to $\mathcal{E}$'s challenger. When $\mathcal{E}$ receives c^* from its challenger, it runs $\mathcal{A}$ on this input, where Dec-Oracle calls are again forwarded, and outputs the output bit of $\mathcal{A}$.

Putting Equations 5-10 together, we obtain Equation 4.

From Eq. 4 to Eq. 3. Let t denote the runtime of $\mathcal{A}$ and t' the maximal runtime of the experiment $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\cdot}(\lambda, k)$ involving an arbitrary adversary with runtime t. Furthermore, note that the reduction algorithms $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$ are uniform in the sense that they perform the same operations for any given adversary $\mathcal{A}$ of runtime t. Let s_1, s_2, s_3, and s_4 denote the maximal runtime of our PRG, IND-CCA, PPRF, and $i\mathcal{O}$ reduction algorithm, respectively, for an RPKE adversary with runtime t. As all these reduction algorithms basically execute the R-IND-CCA experiment (including minor modifications) with the RPKE adversary, we have that $t' \approx s_1 \approx s_2 \approx s_3 \approx s_4$. Clearly, the runtime of our reduction algorithms are upper bounded by the corresponding values t_i and thus it follows

$$\begin{aligned} \mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}},\mathcal{A}}(\lambda, k) \quad &\leq \tfrac{1}{2^\lambda} + \mathsf{CAdv}^{\mathsf{prg}}_{\mathsf{PRG}}(s_1, \lambda) + \mathsf{CAdv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}}}(s_2(\lambda, k), k) \\ &\quad + \mathsf{CAdv}^{\mathsf{pprf}}_{\mathsf{F}}(s_3(\lambda, k), \lambda, k) + \mathsf{CAdv}^{\mathsf{io}}_{i\mathcal{O}}(s_4, \lambda + k). \end{aligned} \tag{11}$$

Finally, since the same upper bound (on the right-hand side of Eq. 11) on the advantage holds for any adversary $\mathcal{A}$ with runtime t, this is also an upper bound for $\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}}}(t, \lambda, k)$. □

Theorem 4.2. *Let us assume that for any polynomial $s(\ell)$, the concrete advantages $\mathsf{CAdv}^{\mathsf{prg}}_{\mathsf{PRG}}(s(\ell), \ell)$, $\mathsf{CAdv}^{\mathsf{io}}_{i\mathcal{O}}(s(\ell), \ell)$, $\mathsf{CAdv}^{\mathsf{pprf}}_{\mathsf{F}}(s(\ell), \ell)$ and $\mathsf{CAdv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}}}(s(\ell), \ell)$ are negligible. Then $\mathsf{RPKE}_{i\mathcal{O}}$ is R-IND-CCA secure.*

Proof. Let $t(\lambda, k)$ be a polynomial and let us consider the upper bound on $\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}}}(t(\lambda, k), \lambda, k)$ given by Lemma 4.1. First, note that since RPKE is efficient there is also a polynomial bound $t'(\lambda, k)$ on the runtime complexity of the experiment and thus $s_1(\lambda, k)$, $s_2(\lambda, k)$, $s_3(\lambda, k)$, and $s_4(\lambda, k)$ will be polynomial as $t'(\lambda, k) \approx s_1(\lambda, k) \approx s_2(\lambda, k) \approx s_3(\lambda, k) \approx s_4(\lambda, k)$ for all $\lambda, k \in \mathbb{N}$. Furthermore, let $t_1(\lambda, k) := s_1(\lambda, k)$, $t_2(\lambda, k) := s_2(\lambda, k)$, and $t_3(\lambda, k)$ be a polynomial upper bound on $s_3(\lambda, k)$ and $s_4(\lambda, k)$. Now, consider the following partition of $\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{i\mathcal{O}}}(t(\lambda, k), \lambda, k)$ as demanded in Definition 3.2: $f_1(t_1(\lambda, k), \lambda) := \frac{1}{2^\lambda} + \mathsf{CAdv}^{\mathsf{prg}}_{\mathsf{PRG}}(t_1(\lambda, k), \lambda)$, $f_2(t_2(\lambda, k), k) := \mathsf{CAdv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE}_{\mathsf{CCA}}}(t_2(\lambda, k), k)$, and

$$f_3(t_3(\lambda, k), \lambda, k) := \mathsf{CAdv}^{\mathsf{io}}_{i\mathcal{O}}(t_3(\lambda, k), \lambda + k) + \mathsf{CAdv}^{\mathsf{pprf}}_{\mathsf{F}}(t_3(\lambda, k), \lambda + k).$$

Obviously, for all fixed $k \in \mathbb{N}$, $t_1(\lambda, k)$ is a polynomial in a single variable, namely λ, and thus $f_1(t_1(\lambda, k), \lambda)$ is negligible in λ by assumption. Similarly, for all fixed $\lambda \in \mathbb{N}$, $f_2(t_2(\lambda, k), k)$ is negligible in k by assumption. Moreover, for all fixed $k \in \mathbb{N}$ and for all fixed $\lambda \in \mathbb{N}$, $t_3(\lambda, k)$ becomes a polynomial in λ and in k, respectively, and the advantages $\mathsf{CAdv}^{\mathsf{io}}_{i\mathcal{O}}(t_3(\lambda, k), \lambda + k)$ and $\mathsf{CAdv}^{\mathsf{pprf}}_{\mathsf{F}}(t_3(\lambda, k), \lambda + k)$ are negligible in λ and in k by assumption. □

Versatility of our $i\mathcal{O}$-based construction. As one can easily see, the $i\mathcal{O}$-based construction of an RPKE we presented above is very modular and generic: there was no need to modify the standard cryptosystem (the IND-CCA secure PKE) itself to make it reconfigurable but we just added a component "in front" which fed its key generator with independently-looking randomness. Thus, the same component may be used to make other types of cryptosystems reconfigurable in this sense. Immediate applications would be the construction of an $i\mathcal{O}$-based R-IND-CPA secure RPKE from an IND-CPA secure PKE or of an R-EUF-CMA secure reconfigurable signature scheme (cf. Definition 5.2) from an EUF-CMA secure signature scheme. The construction is also very flexible in the sense that it allows to switch to a completely different IND-CCA secure PKE (or at least to a more secure algebraic structure for the PKE) on-the-fly when the short-term security level k gets increased. One may even use the same long-term keys to generate short-term PKIs for multiple different cryptosystems (e.g., a signature and an encryption scheme) used in parallel. We leave the security analysis of such extended approaches as an open problem.

4.2 Reconfigurable Encryption from SCasc

Our second construction of a R-IND-CPA secure reconfigurable encryption scheme makes less strong assumptions than our construction using $i\mathcal{O}$. Namely, it uses a pairing-friendly group generator $\mathcal{G}$ as introduced in Section 2 and the only assumption is (a suitable variant of) the $\mathcal{SC}_k$-MDDH assumption with respect to $\mathcal{G}$. Our construction is heavily inspired by Regev's lattice-based encryption scheme [18] (in its "dual variant" [13]). However, instead of computing with noisy integers, we perform similar computations "in the exponent". (A similar adaptation of lattice-based cryptographic constructions to a group setting was already undertaken in [8], although with different constructions and for a different purpose.)

A two-parameter variant of the $\mathcal{SC}_k$-MDDH assumption. For our purposes, it will be useful to consider the $\mathcal{SC}_k$-MDDH assumption as an assumption in *two* security parameters, λ and k. Namely, let

$$\mathsf{Adv}^{\mathsf{SC}}_{\mathcal{G},\mathcal{B}}(\lambda,k) := \mathsf{Adv}^{\mathcal{D}_k\text{-MDDH}}_{\mathcal{G},\mathcal{A}}(\lambda)$$

where $\mathcal{D}_k = \mathcal{SC}_k$ as defined by Equation 2 in Section 2. Note that this also defines the concrete advantage $\mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(t,\lambda,k)$ (generically defined in Section 2).

It is not immediately clear how to define asymptotic security with this two-parameter advantage function. To do so, we follow the path taken for our reconfigurable security definition, with λ as a long-term, and k as a short term security parameter: We say that the SCasc assumption holds relative to $\mathcal{G}$ iff $\mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(t,\lambda,k)$ can be split up into three components, as follows. We require that for every polynomial $t = t(\lambda,k)$, there exist nonnegatively-valued functions $f_1 : \mathbb{N}^2 \to \mathbb{R}^+_0, f_2 : \mathbb{N}^2 \to \mathbb{R}^+_0, f_3 : \mathbb{N}^3 \to \mathbb{R}^+_0$ and polynomials $t_1(\lambda,k), t_2(\lambda,k), t_3(\lambda,k)$ such that

$$\mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(t(\lambda,k),\lambda,k) \leq f_1(t_1(\lambda,k),\lambda) + f_2(t_2(\lambda,k),k) + f_3(t_3(\lambda,k),\lambda,k)$$

and the following conditions are satisfied for f_1, f_2, f_3:

- For all $k \in \mathbb{N}$ it holds that $f_1(t_1(\lambda,k),\lambda)$ is negligible in λ
- For all $\lambda \in \mathbb{N}$ it holds that $f_2(t_2(\lambda,k),k)$ is negligible in k
- For all $k \in \mathbb{N}$ it holds that $f_3(t_3(\lambda,k),\lambda,k)$ is negligible in λ
- For all $\lambda \in \mathbb{N}$ it holds that $f_3(t_3(\lambda,k),\lambda,k)$ is negligible in k.

The interpretation is quite similar to reconfigurable security: we view λ (which determines, e.g., the group order) as a long-term security parameter. On the other hand, k determines the concrete computational problem considered in this group, and we thus view k as a short-term security parameter. (For instance, it is conceivable that an adversary may successfully break one computational problem

in a given group, but not a potentially harder problem. Hence, increasing k may be viewed as increasing the security of the system.) It is not hard to show that $\mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(t,\lambda,k)$ holds in the generic group model, although, the usual proof technique only allows for a trivial splitting of the adversarial advantage into the f_1, f_2 and f_3.

Choosing subspace elements. We will face the problem of sampling a vector $[\vec{r}] \in G^{k+1}$ satisfying $\vec{r}^\top \cdot \mathbf{A}_x = \vec{y}^\top$ for given $\mathbf{A}_x \in \mathbb{Z}_p^{(k+1)\times k}$ (of the form of Eq. 2) and $[\vec{y}] \in G^k$. One efficient way to choose a uniform solution $[\vec{r}] = [(r_i)_i]$ is as follows: choose r_1 uniformly, and set $[r_{i+1}] = [y_i] - x \cdot [r_i]$ for $2 \le i \le k+1$.

Our scheme RPKE_{SC}. Now our encryption scheme has message space G_T and is given by the following algorithms:

$\mathsf{Setup}(1^\lambda)$**:** sample $(p, G, g, G_T, e) \leftarrow \mathcal{G}(1^\lambda)$ and return $\mathcal{PP} := (p, G, g, G_T, e)$.

$\mathsf{MKGen}(\mathcal{PP})$**:** sample $x \leftarrow \mathbb{Z}_p$ and return $mpk := [x] \in G$ and $msk := x$.

$\mathsf{CRSGen}(\mathcal{PP}, 1^k)$**:** sample $\vec{y} \leftarrow \mathbb{Z}_p^k$ and return $CRS := (1^k, \mathcal{PP}, [\vec{y}^\top] \in G^k)$.

$\mathsf{PKGen}(CRS, mpk)$**:** compute $[\mathbf{A}_x]$ from $mpk = [x]$ and return $pk := (CRS, [\mathbf{A}_x] \in G^{(k+1)\times k})$.

$\mathsf{SKGen}(CRS, msk)$**:** compute $\mathbf{A}_x$ from $msk = x$ and sample a uniform solution $[\vec{r}] \in G^{k+1}$ of $\vec{r}^\top \cdot \mathbf{A}_x = \vec{y}^\top$, and return $sk := (CRS, [\vec{r}])$.

$\mathsf{Enc}(pk, m)$**:** sample $\vec{s} \leftarrow \mathbb{Z}_p^k$, and return

$$c = ([\vec{R}], [S]_T) = ([\mathbf{A}_x \cdot \vec{s}], [\vec{y}^\top \cdot \vec{s}]_T \cdot m) \in G^{k+1} \times G_T.$$

$\mathsf{Dec}(sk, c)$**:** return $m = [S]_T - [\vec{r}^\top \cdot \vec{R}]_T \in G_T$.

Correctness and security. Correctness follows from

$$\mathsf{Dec}(sk, c) = [S]_T - [\vec{r}^\top \cdot \vec{R}]_T = \left([\vec{y}^\top \cdot \vec{s}]_T - [\vec{r}^\top \cdot \mathbf{A}_x \cdot \vec{s}]_T\right) \cdot m,$$

since $\vec{y}^\top = \vec{r}^\top \cdot \mathbf{A}_x$ by definition. For security, consider

Lemma 4.3. *Let $t \in \mathbb{N}$ be given and let t' denote the maximal runtime of the experiment $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC},\cdot}(\lambda, k)$ involving arbitrary adversaries with runtime t. Then it holds that*

$$\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cpa}}_{\mathsf{RPKE}_{SC}}(t,\lambda,k) \le \frac{1}{2^\lambda} + \mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(s,\lambda,k) \tag{12}$$

where $t' \approx s$.

Proof. Similar to the proof of lemma 4.1, the following reduction will be in the non-uniform setting, where we consider an adversary $\mathcal{A}$ against RPKE_{SC} for fixed security parameters λ and k. We show that $\mathcal{A}$ can be turned into an algorithm $\mathcal{B}$ solving SCasc for fixed λ and k with advantage $\mathsf{Adv}^{\mathsf{SC}}_{\mathcal{G},\mathcal{B}}(\lambda, k)$ such that

$$\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cpa}}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) \le \frac{1}{2^\lambda} + \mathsf{Adv}^{\mathsf{SC}}_{\mathcal{G},\mathcal{B}}(\lambda,k). \tag{13}$$

We proceed in games, with **Game** 1 being the $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cpa}}_{\mathsf{RPKE}_{SC},\mathcal{A}}$ experiment. Let $\mathsf{Adv}^{\mathrm{Game}_i}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k)$ denote the advantage of $\mathcal{A}$ in Game i. Thus, by definition,

$$\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) = \mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k). \tag{14}$$

In **Game** 2, we implement the $\mathsf{Break}(\mathcal{PP}, msk, \cdot)$ oracle differently for $\mathcal{A}$. Namely, recall that in Game 1, upon input $k' < k$, Break chooses a CRS $CRS_{k'} = (1^{k'}, \mathcal{PP}, [\vec{y}^\top] \leftarrow G^{k'})$, then computes a secret key $sk_{k'} = [\vec{r}] \in G^{k'+1}$ with $\vec{r}^\top \mathbf{A}_x = \vec{y}^\top$, and finally returns $CRS_{k'}$ and $sk_{k'}$ to $\mathcal{A}$.

Instead, we will now let Break first choose $\vec{r} \in \mathbb{Z}_p^{k'+1}$ uniformly, and then compute $[\vec{y}^\top] = [\vec{r}^\top \mathbf{A}_x]$ from $\vec{r}$ and set $CRS_{k'} = (1^{k'}, \mathcal{PP}, [\vec{y}^\top])$. This yields exactly the same distribution for $sk_{k'}$ and $CRS_{k'}$, but only requires knowledge about $[\mathbf{A}_x]$ (and not $\mathbf{A}_x$). Hence, we have

$$\mathsf{Adv}^{\mathrm{Game}_1}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) = \mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k). \tag{15}$$

In **Game** 3, we prepare the challenge ciphertext c^* differently for $\mathcal{A}$. As a prerequisite, we let the game also choose CRS^* like the Break oracle from Game 2 chooses the $CRS_{k'}$. In other words, we set up $CRS^* = [\vec{y}^\top] = [\vec{r^*}^\top \mathbf{A}_x]$ for uniformly chosen $\vec{r^*}$. This way, we can assume that $sk^* = (CRS^*, [\vec{r^*}])$ is known to the game, even for an externally given $[\mathbf{A}_x]$.

Next, recall that in Game 2, we have first chosen $\vec{s} \leftarrow \mathbb{Z}_p^k$ and then computed $c^* = ([\vec{R}], [S]_T) = ([\mathbf{A}_x \cdot \vec{s}], [\vec{y}^\top \cdot \vec{s}]_T \cdot m_b)$. In Game 3, we still first choose $\vec{s}$ and compute $[\vec{R}] = [\mathbf{A}_x \cdot \vec{s}]$. However, we then compute $[S]_T = [\vec{r^*}^\top \cdot R]_T \cdot m_b$ in a black-box way from $[\vec{R}]$, without using $\vec{s}$ again.

These changes are again purely conceptual, and we get

$$\mathsf{Adv}^{\mathrm{Game}_2}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) = \mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k). \tag{16}$$

Now, in **Game** 4, we are finally ready to use the SCasc assumption. Specifically, instead of computing the value $[\vec{R}]$ of c^* as $[\vec{R}] = [\mathbf{A}_x \cdot \vec{s}]$ for a uniformly chosen $\vec{s} \in \mathbb{Z}_p^k$, we sample $[\vec{R}] \in G^{k+1}$ independently and uniformly. (By our change from Game 3, then $[S]_T$ is computed from $[\vec{R}]$ using sk^*.)

Our change hence consists in replacing an element of the form $[\mathbf{A}_x \cdot \vec{s}]$ by a random vector of group elements. Besides, at this point, our game only requires knowledge of $[\mathbf{A}_x]$ (but not of $\mathbf{A}_x$). Hence, a straightforward reduction to the SCasc assumption yields an adversary $\mathcal{B}$ with

$$\mathsf{Adv}^{\mathsf{SC}}_{\mathcal{G},\mathcal{B}}(\lambda,k) = \left|\mathsf{Adv}^{\mathrm{Game}_4}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) - \mathsf{Adv}^{\mathrm{Game}_3}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k)\right|. \tag{17}$$

Finally, it is left to observe that in Game 4, the challenge ciphertext is (statistically close to) independently random. Indeed, recall that the challenge ciphertext is chosen as $c^* = ([\vec{R}], [S]_T)$ for uniform $\vec{R} \in \mathbb{Z}_p^{k+1}$, and $[S]_T = [\vec{r^*}^\top \cdot R]_T \cdot m_b$. Suppose now that $\vec{R}$ does not lie in the image of $\mathbf{A}_x$. (That is, $\vec{R}$ cannot be explained as a combination of columns of $\mathbf{A}_x$.) Then, for random $\vec{r}$, the values $\vec{r^*}^\top \mathbf{A}_x$ and $\vec{r^*}^\top \cdot R$ are independently random. In particular, even given $[\mathbf{A}_x]$ and CRS^*, the value $[\vec{r^*}^\top \cdot R]_T$ looks independently random to $\mathcal{A}$.

Hence, $\mathcal{A}$'s view is independent of the encrypted message m_b (at least when conditioned on $\vec{R}$ not being in the image of $\mathbf{A}_x$). On the other hand, since $\vec{R}$ is uniformly random in Game 4, it lies in the image of $\mathbf{A}_x$ only with probability $1/p$. Thus, we get

$$\mathsf{Adv}^{\mathrm{Game}_4}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) \leq \frac{1}{p}. \tag{18}$$

Putting Eq. 14-18 together (and using that $p \geq 2^\lambda$), we obtain Equation 13.

From Eq. 13 to Eq. 12. Let t denote the runtime of $\mathcal{A}$ and t' the maximal runtime of the experiment $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC},\cdot}(\lambda,k)$ involving an arbitrary adversary with runtime t. Note that the reduction algorithm $\mathcal{B}$ is uniform in the sense that it performs the same operations for any given adversary $\mathcal{A}$ of runtime t. Let s denote the maximal runtime of our SCasc algorithm for an RPKE adversary with runtime t. As the SCasc algorithm basically executes the R-IND-CCA experiment (including minor modifications) with the RPKE adversary, we have that $t' \approx s$. Clearly, the runtime of $\mathcal{B}$ is upper bounded by s and thus it follows

$$\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC},\mathcal{A}}(\lambda,k) \leq \frac{1}{2^\lambda} + \mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(s,\lambda,k). \tag{19}$$

Finally, since the same upper bound (on the right-hand side of Eq. 19) on the advantage holds for any adversary $\mathcal{A}$ with runtime t, this is also an upper bound for $\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC}}(t, \lambda, k)$. □

Theorem 4.4. *If the two-parameter variant of the SCasc assumption holds, then* RPKE_{SC} *is R-IND-CPA secure.*

Proof. Let $t(\lambda, k)$ be a polynomial. Since RPKE_{SC} is efficient, $t'(\lambda, k)$ will be polynomial and so $s(\lambda, k)$. As $s(\lambda, k)$ is polynomial, according to the SCasc assumption there exist functions g_1, g_2, and g_3 as well as polynomials $s_1(\lambda, k)$, $s_2(\lambda, k)$, and $s_3(\lambda, k)$ such that

$$\mathsf{CAdv}^{\mathsf{SC}}_{\mathcal{G}}(s(\lambda, k), \lambda, k) \leq g_1(s_1(\lambda, k), \lambda) + g_2(s_2(\lambda, k), k) + g_3(s_3(\lambda, k), \lambda, k).$$

Now, consider the following partitioning of $\mathsf{CAdv}^{\mathsf{r\text{-}ind\text{-}cca}}_{\mathsf{RPKE}_{SC}}(t(\lambda, k), \lambda, k)$ as specified in Definition 3.2: $f_1(s_1(\lambda, k), \lambda) := \frac{1}{2^\lambda} + g_1(s_1(\lambda, k), \lambda, k)$, $f_2(s_2(\lambda, k), k) := g_2(s_2(\lambda, k), \lambda, k)$, and $f_3(s_3(\lambda, k), \lambda, k) = g_3(s_3(\lambda, k), \lambda, k)$. Clearly, the properties demanded for f_1, f_2, f_3 by Definition 3.2 immediately follow from the SCasc assumption. □

5 Reconfigurable Signatures

The concept of reconfiguration is not restricted to encryption schemes. In this section, we consider the case of reconfigurable signatures. We start with some preliminaries, define reconfigurable signatures and a security experiment (both in line with the encryption case) and finally give a construction.

5.1 Preliminaries

Signature schemes. A signature scheme SIG with message space $\mathcal{M}$ consists of three PPT algorithms $\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}$. $\mathsf{Setup}(1^\lambda)$ outputs public parameters $\mathcal{PP}$ for the scheme. Key generation $\mathsf{Gen}(\mathcal{PP})$ outputs a verification key vk and a signing key sk. The signing algorithm $\mathsf{Sig}(sk, m)$ takes the signing key and a message $m \in \mathcal{M}$, and outputs a signature σ. Verification $\mathsf{Ver}(vk, \sigma, m)$ takes the verification key, a signature and a message m and outputs 1 or $\perp$. For correctness, we require that for all $m \in \mathcal{M}$ and all $(vk, sk) \leftarrow \mathsf{Gen}(1^k)$ we have $\mathsf{Ver}(sk, \mathsf{Sig}(sk, m), m) = 1$.

EUF-CMA security. The *EUF-CMA-advantage* of an adversary $\mathcal{A}$ on SIG is defined by $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = 1]$ for the experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}$ described below. In $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}$, first, $\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(pk, sk) \leftarrow \mathsf{Gen}(\mathcal{PP})$ is sampled. The we run $\mathcal{A}$ on input pk, where $\mathcal{A}$ also has access to a signature oracle. The experiment returns 1 if for $\mathcal{A}$'s output (σ^*, m^*) it holds that $\mathsf{Ver}(pk, \sigma^*, m^*) = 1$ and m^* was not sent to the signature oracle. A signature scheme SIG is called EUF-CMA-secure if for all PPT algorithms $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ is negligible.

Non-interactive proof systems. A non-interactive proof system for a language $\mathcal{L}$ consists of three PPT algorithms $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$. $\mathsf{CRSGen}(\mathcal{L})$ gets as input information about the language and outputs a *common reference string* (CRS). $\mathsf{Prove}(CRS, x, w)$ with statement x and witness w outputs a proof π, and $\mathsf{Ver}(CRS, \pi, x)$ outputs 1 if π is a valid proof for $x \in \mathcal{L}$, and $\perp$ otherwise. The proof system is *complete* if Ver always accepts proofs if x is contained in $\mathcal{L}$, and it is *perfectly sound* if Ver always rejects proofs if x is not in $\mathcal{L}$.

Witness indistinguishability (WI). Suppose a statement $x \in \mathcal{L}$ has more than one witness. A proof of membership can be generated using any of the witnesses. If a proof $\pi \leftarrow \mathsf{Prove}(CRS, x, w)$ information theoretically hides the choice of the witness, it is called *perfectly witness indistinguishable.*

Groth-Sahai (GS) proofs. In [15], Groth and Sahai introduced efficient non-interactive proof systems in pairing-friendly groups. We will only give a high level overview of the properties that are

needed for our reconfigurable signature scheme and refer to the full version [15] for the details of their construction.

In GS proof systems, the algorithm CRSGen takes as input a pairing-friendly group $\mathbb{G} := (p, G, g, G_T, e)$ and outputs a CRS suitable for proving satisfiability of various types of equations in these groups. Furthermore, CRSGen has two different modes of operation, producing a CRS that leads to either perfectly witness indistinguishable or perfectly sound proofs. The two types of CRS can be shown to be computationally indistinguishable under different security assumptions such as subgroup decision, SXDH and 2-Linear.

In both modes, CRSGen additionally outputs a trapdoor. In the WI mode, this trapdoor can be used to produce proofs of false statements[6]. In the sound mode, the trapdoor can be used to extract the witness from the proof. To easily distinguish the two operating modes, we equip CRSGen with an additional parameter $mode \in \{\mathtt{wi}, \mathtt{sound}\}$.

Statements provable with GS proofs have to be formulated in terms of satisfiability of equations in pairing-friendly groups. For example, it is possible to prove the statement $\mathcal{X} :=$ "$\exists s \in \mathbb{Z}_n : [s]_1 = S$" for an element $S \in G_1$. A witness for this statement is a value s satisfying the equation $[s] = S$, i.e., the DL of S to the basis g_1. Furthermore, GS proofs are nestable and thus admit proving statements about proofs, e.g., $\mathcal{Y} :=$ "$\exists \pi : \mathsf{Ver}(CRS, \pi, \mathcal{X}) = 1$".

5.2 Definitions

Similar to the case of RPKE, we can define reconfigurable signatures.

Definition 5.1. *A reconfigurable signature (RSIG) scheme* RSIG *consists of algorithms* Setup, MKGen, CRSGen, PKGen, SKGen, Sig *and* Ver. *The first five algorithms are defined as in Definition 3.1.* Sig *and* Ver *are the signature generation and verification algorithms and are defined as in a regular signature scheme.* RSIG *is called correct if for all* $\lambda, k \in \mathbb{N}$, $\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$, $(mpk, msk) \leftarrow \mathsf{MKGen}(\mathcal{PP})$, $CRS \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$, *messages* $m \in \mathcal{M}$, $sk \leftarrow \mathsf{SKGen}(CRS, msk)$ *and* $pk \leftarrow \mathsf{PKGen}(CRS, mpk)$ *we have that* $\mathsf{Ver}(pk, \mathsf{Sig}(sk, m), m) = 1$.

We define R-EUF-CMA security for an RSIG scheme RSIG analogously to R-IND-CCA security for RPKE, where the security experiment $\mathsf{Exp}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}}(\lambda,\mathrm{k})$ is defined in Figure 3.

Definition 5.2. *Let* RSIG *be an RSIG scheme according to Definition 5.1. Then we define the advantage of an adversary* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}}(\lambda, k) := \Pr[\mathsf{Exp}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}}(\lambda, k) = 1]$$

where $\mathsf{Exp}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}}(\lambda, k)$ *is the experiment given in Figure 3. The concrete advantage* $\mathsf{CAdv}^{\text{r-euf-cma}}_{\mathsf{RSIG}}(t, \lambda, k)$ *of adversaries against* RSIG *with time complexity* t *follows canonically (cf. Section 2).*

An RSIG scheme RSIG *is then called R-EUF-CMA secure if for all polynomials* $t(\lambda, k)$, *there exist positive functions* $f_1 : \mathbb{N}^2 \to \mathbb{R}_0^+$, $f_2 : \mathbb{N}^2 \to \mathbb{R}_0^+$, *and* $f_3 : \mathbb{N}^3 \to \mathbb{R}_0^+$ *as well as polynomials* $t_1(\lambda, k)$, $t_2(\lambda, k)$, *and* $t_3(\lambda, k)$ *such that*

$$\mathsf{CAdv}^{\text{r-euf-cma}}_{\mathsf{RSIG}}(t(\lambda, k), \lambda, k) \leq f_1(t_1(\lambda, k), \lambda) + f_2(t_2(\lambda, k), k) + f_3(t_3(\lambda, k), \lambda, k)$$

for all λ, k, *and the following conditions are satisfied for* f_1, f_2, f_3*:*

- *For all* $k \in \mathbb{N}$ *it holds that* $f_1(t_1(\lambda, k), \lambda)$ *is negligible in* λ

[6] Actually, the original paper only describes a method for generating proofs for specific false statements. Arbitrary statements can be proven at the cost of slightly larger proofs and CRSs, using known methods that apply to WI proofs [14].

Experiment $\mathsf{Exp}^{\mathsf{r\text{-}euf\text{-}cma}}_{\mathsf{RSIG},\mathcal{A}}(\lambda,\mathrm{k})$

$\mathcal{PP} \leftarrow \mathsf{Setup}(1^\lambda)$

$(mpk, msk) \leftarrow \mathsf{MKGen}(\mathcal{PP})$

$state \leftarrow \mathcal{A}^{\mathsf{Break}(\mathcal{PP},msk,\cdot)}(1^\lambda, 1^k, \mathcal{PP}, mpk, \text{“learn”})$

$CRS^* \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$

$sk^* \leftarrow \mathsf{SKGen}(CRS^*, msk)$

$pk^* \leftarrow \mathsf{PKGen}(CRS^*, mpk)$

$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sig}(sk^*,\cdot)}(CRS^*, state)$

Let $k_1, \ldots, k_\ell$ be the inputs sent to the Break-Oracle by $\mathcal{A}$. On input k_i, the Break-Oracle returns $CRS_{k_i} \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^{k_i})$ as well as $sk_{k_i} \leftarrow \mathsf{SKGen}(CRS_{k_i}, msk)$ to $\mathcal{A}$.

Return 1 if $k_i < k$ for all i, $\mathsf{Ver}(pk^*, \sigma^*, m^*) = 1$, and m^* was not an input to the Sig-Oracle. Otherwise, return 0.

Figure 3: R-EUF-CMA experiment for a reconfigurable signature scheme RSIG.

- *For all* $\lambda \in \mathbb{N}$ *it holds that* $f_2(t_2(\lambda, k), k)$ *is negligible in* k
- *For all* $k \in \mathbb{N}$ *it holds that* $f_3(t_3(\lambda, k), \lambda, k)$ *is negligible in* λ
- *For all* $\lambda \in \mathbb{N}$ *it holds that* $f_3(t_3(\lambda, k), \lambda, k)$ *is negligible in* k

5.3 Reconfigurable Signatures from Groth-Sahai Proofs

The intuition behind our scheme is as follows. Each user of the system has a long-term key pair, consisting of a public instance of a hard problem and a private solution of this instance. A valid signature is a proof of knowledge of *either* knowledge of the long-term secret key *or* a valid signature of the message under another signature scheme. The proof system and signature scheme for generating the proofs of knowledge are published, e.g. using a CRS. We are now able to reconfigure the scheme by pdating the CRS with a new proof system and a new signature scheme. This way, old short-term secret keys of a user (i.e., valid proofs of knowledge of the user's long-term secret key under deprecated proof systems) become useless and can thus be leaked to the adversary.

Our reconfigurable signature scheme RSIG has message space $\mathcal{M} = \{0,1\}^m$. It makes use of a symmetric pairing-friendly group generator $\mathcal{G}$, a family of GS proof systems $\mathsf{PS} := \{\mathsf{PS}_k := (\mathsf{CRSGen}_{\mathsf{PS}_k}, \mathsf{Prove}_{\mathsf{PS}_k}, \mathsf{Ver}_{\mathsf{PS}_k})\}_{k\in\mathbb{N}}$ for proving equations in the groups generated by $\mathcal{G}(1^\lambda)$ and a family of EUF-CMA-secure signature schemes $\mathsf{SIG} := \{\mathsf{SIG}_k := (\mathsf{Setup}_{\mathsf{SIG}_k}, \mathsf{Gen}_{\mathsf{SIG}_k}, \mathsf{Sig}_{\mathsf{SIG}_k}, \mathsf{Ver}_{\mathsf{SIG}_k})\}_{k\in\mathbb{N}}$ with message space $\mathcal{M}$, where $\mathsf{Setup}_{\mathsf{SIG}_k}(1^\lambda)$ outputs $\mathbb{G}$ with $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$ for all $k \in \mathbb{N}$ (i.e., each SIG_k can be instantiated using the same symmetric pairing-friendly groups $\mathbb{G}$).

Two-parameter families of GS proofs and EUF-CMA-secure signatures. Let us view PS as a family of GS proof systems and SIG a family of EUF-CMA-secure signature schemes defined in *two* security parameters λ and k. Such families may be constructed based on the (two parameters variant) of the SCasc assumption or other matrix assumptions. Consequently, we consider a security experiment where the adversary receives two security parameters and has advantage $\mathsf{Adv}^{\mathsf{ind\text{-}crs}}_{\mathsf{PS},\mathcal{A}}(\lambda, k)$ and $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{B}}(\lambda, k)$, respectively. Note that this also defines the concrete advantages $\mathsf{CAdv}^{\mathsf{ind\text{-}crs}}_{\mathsf{PS}}(t, \lambda, k)$ and $\mathsf{CAdv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}(t, \lambda, k)$ (as generically defined in Section 2). We define asymptotic security for these families following the approach taken for our reconfigurable security definition. That means, we call PS (SIG) secure if for every polynomial $t(\lambda, k)$ the advantage $\mathsf{CAdv}^{\mathsf{ind\text{-}crs}}_{\mathsf{PS}}(t(\lambda, k), \lambda, k)$ ($\mathsf{CAdv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}}(t(\lambda, k), \lambda, k)$) can be split up into nonnegatively-valued functions

$\underline{\mathsf{Setup}(1^\lambda)}$	$\underline{\mathsf{MKGen}(\mathcal{PP})}$
$(p, G, g, G_T, e) \leftarrow \mathcal{G}(1^\lambda)$ return $\mathcal{PP} := (p, G, g, G_T, e)$	parse $\mathcal{PP} := (p, G, g, G_T, e)$ $x, y \leftarrow \mathbb{Z}_n$ return $mpk := ([x], [y])$, $msk := [xy]$
$\underline{\mathsf{CRSGen}(\mathcal{PP}, 1^k)}$ $(CRS_{\mathsf{PS}_k}, td_k) \leftarrow \mathsf{CRSGen}_{\mathsf{PS}_k}(\mathtt{wi}, \mathcal{PP})$ $(\widetilde{sk}_k, \widetilde{vk}_k) \leftarrow \mathsf{Gen}_{\mathsf{SIG}_k}(\mathcal{PP})$ return $CRS_k := (CRS_{\mathsf{PS}_k}, \widetilde{vk}_k, \mathcal{PP}, k)$	
$\underline{\mathsf{PKGen}(CRS_k, mpk)}$ return $pk_k := (CRS_k, mpk)$	$\underline{\mathsf{SKGen}(CRS_k, msk)}$ parse CRS_k as $(CRS_{\mathsf{PS}_k}, \widetilde{vk}_k, \mathcal{PP}, k)$ set $\mathcal{X} :=$ "$\exists z : e(mpk_1, mpk_2) = e(z, [1])$" $\pi_k \leftarrow \mathsf{Prove}_{\mathsf{PS}_k}(CRS_{\mathsf{PS}_k}, \mathcal{X}, msk)$ return $sk_k := (CRS_k, \pi_k)$
$\underline{\mathsf{Sig}(m, sk_k)}$ parse sk_k as (CRS_k, π_k) and CRS_k as $(CRS_{\mathsf{PS}_k}, \widetilde{vk}_k, \mathcal{PP}, k)$ set $\mathcal{Y}_k :=$ "$\exists(\pi_k, \Sigma_k) : \mathsf{Ver}_{\mathsf{PS}_k}(CRS_{\mathsf{PS}_k}, \pi_k, \mathcal{X}) = 1 \vee \mathsf{Ver}_{\mathsf{SIG}_k}(\widetilde{vk}_k, \Sigma_k, m) = 1$" $\pi_m \leftarrow \mathsf{Prove}_{\mathsf{PS}_k}(CRS_{\mathsf{PS}_k}, \mathcal{Y}_k, sk_k)$ return $\sigma := (\pi_m, \mathcal{Y}_k)$	$\underline{\mathsf{Ver}(pk_k, \sigma, m)}$ parse pk_k as (CRS_k, mpk) and CRS_k as $(CRS_{\mathsf{PS}_k}, \widetilde{vk}_k, \mathcal{PP}, k)$ parse $\sigma := (\pi_m, \mathcal{Y}_k)$ verify that $\mathcal{Y}_k$ contains m and $\mathcal{X}$ and $\mathcal{X}$ contains mpk return $\mathsf{Ver}_{\mathsf{PS}_k}(CRS_{\mathsf{PS}_k}, \pi_m, \mathcal{Y}_k)$

Figure 4: Our reconfigurable signature scheme

$f_1 : \mathbb{N}^2 \to \mathbb{R}_0^+, f_2 : \mathbb{N}^2 \to \mathbb{R}_0^+, f_3 : \mathbb{N}^3 \to \mathbb{R}_0^+$ such that for some polynomials $t_1(\lambda, k)$, $t_2(\lambda, k)$, $t_3(\lambda, k)$ the sum $f_1(t_1(\lambda, k), \lambda) + f_2(t_2(\lambda, k), k) + f_3(t_3(\lambda, k), \lambda, k)$ is an upper bound on the advantage. Furthermore, the following conditions need to be satisfied for f_1, f_2, f_3:

- For all $k \in \mathbb{N}$ it holds that $f_1(t_1(\lambda, k), \lambda)$ is negligible in λ
- For all $\lambda \in \mathbb{N}$ it holds that $f_2(t_2(\lambda, k), k)$ is negligible in k
- For all $k \in \mathbb{N}$ it holds that $f_3(t_3(\lambda, k), \lambda, k)$ is negligible in λ
- For all $\lambda \in \mathbb{N}$ it holds that $f_3(t_3(\lambda, k), \lambda, k)$ is negligible in k.

Correctness of RSIG, in terms of Definition 5.1, directly follows from the completeness of the underlying proof system.

Lemma 5.3. *Let a $t \in \mathbb{N}$ be given and let t' denote the maximal runtime of the experiment $\mathsf{Exp}^{\mathsf{r\text{-}euf\text{-}cma}}_{\mathsf{RSIG},\cdot}(\lambda, k)$ involving arbitrary adversaries with runtime t. Then it holds that*

$$\mathsf{CAdv}^{\mathsf{r\text{-}euf\text{-}cma}}_{\mathsf{RSIG}}(t, \lambda, k) \leq 2 \cdot \mathsf{CAdv}^{\mathit{ind\text{-}crs}}_{\mathsf{PS}}(s_1, \lambda, k) + \mathsf{CAdv}^{\mathit{cdh}}_{\mathcal{G}}(s_2, \lambda) + \mathsf{CAdv}^{\mathit{euf\text{-}cma}}_{\mathsf{SIG}}(s_3, \lambda, k) \tag{20}$$

where $t' \approx s_1 \approx s_2 \approx s_3$.

Theorem 5.4. *Let us assume that PS is a secure two-parameter family of Groth-Sahai proof systems, SIG a secure two-parameter family of EUF-CMA secure signature schemes and the CDH assumption holds with respect to $\mathcal{G}$. Then RSIG is R-EUF-CMA secure.*

We omit the proof of Theorem 5.4 as it is analogous to the proof of Lemma 4.4. In the remainder of this section, we sketch a proof for Lemma 5.3.

Proof sketch: We use a hybrid argument to prove our theorem. Starting with the R-EUF-CMA security game, we end up with a game in which the adversary has no chance of winning. It follows that $\mathsf{Adv}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)$ is smaller than the sum of advantages of adversaries distinguishing between all subsequent intermediate games. Throughout the proof, $\mathsf{Adv}^{Gi}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)$ denotes the winning probability of $\mathcal{A}_{\mathsf{RSIG}}$ when running in game i.

Game 0: This is the original security game $\mathsf{Exp}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}_{\mathsf{RSIG}}}$. Note that the signature oracle of $\mathcal{A}_{\mathsf{RSIG}}$ is implemented using sk_k and thus, implicitly, msk as a witness. We have that

$$\mathsf{Adv}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) = \mathsf{Adv}^{G0}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) \tag{21}$$

Game 1: Here we modify the implementation of the signature oracle by letting the experiment use the formerly unused signing key of the signature scheme SIG_k. More formally, let *state* denote the output of $\mathcal{A}_{\mathsf{RSIG}}{}^{\mathsf{Break}}(\mathcal{PP}, mpk, \text{“learn”})$. While running $(CRS^*, \widetilde{vk}^*, \mathcal{PP}, k) \leftarrow \mathsf{CRSGen}(\mathcal{PP}, 1^k)$, the experiment learns $\widetilde{sk}^*$, the signing key corresponding to $\widetilde{vk}^*$. We now let the experiment answer $\mathcal{A}_{\mathsf{RSIG}}$'s oracle queries $\mathsf{Sig}_k(sk^*, m)$ for $m \in \mathcal{M}$ with signatures $\mathsf{Prove}_{\mathsf{PS}_k}(CRS^*, \mathcal{Y}^*, \tau)$, where $\tau \leftarrow \mathsf{Sig}_{\mathsf{SIG}_k}(\widetilde{sk}^*, m)$ and $\mathcal{Y}^* := \text{”}\exists(\pi^*, \Sigma^*) : \mathsf{Ver}_{\mathsf{PS}_k}(CRS^*, \pi^*, \mathcal{X}) = 1 \vee \mathsf{Ver}_{\mathsf{SIG}_k}(\widetilde{vk}^*, \Sigma^*, m) = 1\text{”}$.

Since the proofs generated by PS_k are perfectly WI, the $\mathcal{A}_{\mathsf{RSIG}}$'s view in game 0 and game 1 is exactly the same and thus we have $\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) = \mathsf{Adv}^{G0}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)$

Game 2: In this game, we want to switch the CRS for which $\mathcal{A}_{\mathsf{RSIG}}$ forges a message from witness indistinguishable to sound mode. For this, the experiment runs $(CRS_{\mathsf{PS}_k}, td_k) \leftarrow \mathsf{CRSGen}_{\mathsf{PS}_k}(\mathtt{sound}, \mathcal{PP})$ and $(\widetilde{sk}^*, \widetilde{vk}^*) \leftarrow \mathsf{Gen}_{\mathsf{SIG}_k}(\mathcal{PP})$ and sets $CRS^* := (CRS_{\mathsf{PS}_k}, \widetilde{vk}^*, \mathcal{PP}, k)$.

Claim 5.5. *For every λ and k and adversary $\mathcal{A}_{\mathsf{RSIG}}$, there is an adversary $\mathcal{A}_{\mathsf{PS}_k}$ with $\mathbf{T}(\mathcal{A}_{\mathsf{RSIG}}) \approx \mathbf{T}(\mathcal{A}_{\mathsf{PS}_k})$ and*

$$\mathsf{Adv}^{\textit{ind-crs}}_{\mathsf{PS},\mathcal{A}_{\mathsf{PS}_k}}(\lambda,k) := \left|\frac{1}{2} - \Pr\left[\mathcal{A}_{\mathsf{PS}_k}(CRS_{\mathsf{PS}_k}) \to \mathtt{mode}\right]\right| = \left|\frac{\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)}{2}\right| \tag{22}$$

where $(CRS_{\mathsf{PS}_k}, td_k) \leftarrow \mathsf{CRSGen}_{\mathsf{PS}_k}(\mathtt{mode}, \mathcal{PP})$ and $\mathtt{mode} \in \{\mathtt{wi}, \mathtt{sound}\}$.

Proof. Note that $\mathcal{A}_{\mathsf{RSIG}}$'s view in game 1 and 2 is exactly the same until he sees CRS^*. We construct $\mathcal{A}_{\mathsf{PS}_k}$ as follows. $\mathcal{A}_{\mathsf{PS}_k}$ gets CRS_{PS_k} and then plays game 1 with $\mathcal{A}_{\mathsf{RSIG}}$ until $\mathcal{A}_{\mathsf{RSIG}}$ outputs *state*. Now $\mathcal{A}_{\mathsf{PS}_k}$ sets $CRS^* := (CRS_{\mathsf{PS}_k}, \widetilde{vk}^*, \mathcal{PP}, k)$ and proceeds the game. Note that this is possible since $\mathcal{A}_{\mathsf{PS}_k}$ does not make use of a trapdoor for CRS_{PS_k}. $\mathcal{A}_{\mathsf{PS}_k}$ finally outputs `wi` if $\mathcal{A}_{\mathsf{RSIG}}$ wins the game. If $\mathcal{A}_{\mathsf{RSIG}}$ loses, $\mathcal{B}$ outputs `sound`.

We now analyze the advantage of $\mathcal{A}_{\mathsf{PS}_k}$ in guessing the CRS mode. For this, note that if $\mathtt{mode} = \mathtt{wi}$, then $\mathcal{A}_{\mathsf{RSIG}}$'s view is as in game 1, and if $\mathtt{mode} = \mathtt{sound}$, then $\mathcal{A}_{\mathsf{RSIG}}$'s view is as in game 2. Let X_i

denote the event that $\mathcal{A}_{\mathsf{RSIG}}$ wins game i, and thus $\mathsf{Adv}^{Gi}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) = \Pr[X_i]$. We have that

$$
\begin{aligned}
&\Pr[\mathcal{A}_{\mathsf{PS}_k}\text{wins}] = \Pr[\mathcal{A}_{\mathsf{PS}_k} \text{ wins } |\mathtt{mode} = \mathtt{wi}] + \Pr[\mathcal{A}_{\mathsf{PS}_k} \text{ wins } |\mathtt{mode} = \mathtt{sound}] \\
=&\frac{1}{2}\sum_{i=1}^{2}(\Pr[\mathcal{A}_{\mathsf{PS}_k} \text{ wins } |X_i] + \Pr[\mathcal{A}_{\mathsf{PS}_k} \text{ wins } |\neg X_i]) \\
=&\frac{1}{2}(1\cdot \mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) + 0\cdot(1-\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)) + 0\cdot \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) + 1\cdot(1-\mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)) \\
=&\frac{1}{2}(\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) + 1 - \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)) = \frac{1}{2} + \frac{\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)}{2} \\
\Rightarrow &\Pr[\mathcal{A}_{\mathsf{PS}_k}\text{wins}] - \frac{1}{2} = \frac{\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)}{2} \\
\Rightarrow &\left|\Pr[\mathcal{A}_{\mathsf{PS}_k}\text{wins}] - \frac{1}{2}\right| = \left|\frac{\mathsf{Adv}^{G1}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)}{2}\right|
\end{aligned}
$$

□

Game 3: Now, the experiment no longer uses knowledge of msk to produce answers $sk_k \leftarrow \mathsf{SKGen}(CRS_{\mathsf{PS}_k}, msk)$ to Break-queries. Instead, we let the experiment use the trapdoor of the CRS to generate the proofs. This can be done since the experiment always answers Break-oracle queries by running $(CRS_{\mathsf{PS}_k}, td_k) \leftarrow \mathsf{CRSGen}_{\mathsf{PS}_k}(\mathtt{wi}, \mathcal{PP})$ and, since in wi mode, td_k can be used to simulate a proof sk_k without actually using msk. Moreover, the proofs are perfectly indistinguishable from the proofs in Game 2 and thus $\mathcal{A}_{\mathsf{RSIG}}$'s view in Games 2 and 3 are identical and we have $\mathsf{Adv}^{G3}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) = \mathsf{Adv}^{G2}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)$

Game 4: We modify the winning conditions of the experiment: $\mathcal{A}_{\mathsf{RSIG}}$ loses if sk^*, i.e., a solution to a CDH instance, can be extracted from the forgery.

Claim 5.6. *For every λ and k, and every adversary $\mathcal{A}_{\mathsf{RSIG}}$, there exists a $\mathcal{A}_{\mathcal{G}}$ with $\mathbf{T}(\mathcal{A}_{\mathsf{RSIG}}) \approx \mathbf{T}(\mathcal{A}_{\mathcal{G}})$ and*

$$\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{G},\mathcal{A}_{\mathcal{G}}}(\lambda) := \Pr[\mathcal{A}_{\mathcal{G}}(\mathbb{G},[x],[y]) = [xy]] \geq \left|\mathsf{Adv}^{G3}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G4}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k)\right| \tag{23}$$

where $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$ and the probability is over the random coins of $\mathcal{G}$ and $\mathcal{A}_{\mathcal{G}}$.

Proof. First note that $\mathcal{A}_{\mathsf{RSIG}}$'s view is identical in both games, since we only modified the winning condition. Let E denote the event that sk^* can be extracted from the forgery produced by $\mathcal{A}_{\mathsf{RSIG}}$. Let X_3, X_4 denote the random variables describing the output of the experiment in Game 3 and Game 4, respectively. From the definition of the winning conditions of both games it follows that

$$
\begin{aligned}
\Pr[X_3 = 1|\neg E] = \Pr[X_4 = 1|\neg E] \Longrightarrow |\Pr[X_3 = 1] - \Pr[X_4 = 1]| &\leq \Pr[E] \\
&\leq \Pr[\mathcal{A}_{\mathcal{G}}(\mathbb{G}, mpk) = msk]
\end{aligned}
$$

where the first inequality follows from the difference lemma [22] and the latter holds because, since msk is not needed to run the experiment, $\mathcal{A}_{\mathcal{G}}$ can run $\mathcal{A}_{\mathsf{RSIG}}$ and, since E happened, extract the CDH solution from the forgery. □

Game 5: We again modify the winning conditions of $\mathcal{A}_{\mathsf{RSIG}}$ by: $\mathcal{A}_{\mathsf{RSIG}}$ loses the game if a valid signature under SIG_k can be extracted from the forgery.

Claim 5.7. *For every λ and k, and every adversary $\mathcal{A}_{\mathsf{RSIG}}$, there exists a $\mathcal{A}_{\mathsf{SIG}_k}$ with $\mathbf{T}(\mathcal{A}_{\mathsf{RSIG}}) \approx \mathbf{T}(\mathcal{A}_{\mathsf{SIG}_k})$ and*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}_k,\mathcal{A}_{\mathsf{SIG}_k}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}_k,\mathcal{A}_{\mathsf{SIG}_k}}(\lambda) = 1\right] \geq \mathsf{Adv}^{G4}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) - \mathsf{Adv}^{G5}_{\mathcal{A}_{\mathsf{RSIG}}}(\lambda,k) \tag{24}$$

Proof. The proof proceeds similar to the proof of the last claim. Note that the signature oracle provided by the EUF-CMA experiment can be used to answer $\mathcal{A}_{\mathsf{RSIG}}$'s queries to the oracle $\mathsf{Sig}_k(sk^*, \cdot)$. □

So far, we made the following changes to the original security experiment.

Game 0: the $\mathsf{Exp}^{\text{r-euf-cma}}_{\mathsf{RSIG},\mathcal{A}_{\mathsf{RSIG}}}$ security experiment

Game 1: experiment uses signatures from SIG_k to implement $\mathsf{Sig}_k(sk^*, \cdot)$

Game 2: experiment switches last CRS to `sound` mode

Game 3: experiment simulates answers to `Break`-queries (from here, the experiment does not need to know msk)

Game 4: $\mathcal{A}_{\mathsf{RSIG}}$ loses if he solves a CDH instance

Game 5: $\mathcal{A}_{\mathsf{RSIG}}$ loses if he breaks EUF-CMA security

Now let us determine the chances of $\mathcal{A}_{\mathsf{RSIG}}$ in winning game 5. If $\mathcal{A}_{\mathsf{RSIG}}$ does not know any of the two witnesses, it follows from the perfect soundness of CRS^* that $\mathcal{A}_{\mathsf{RSIG}}$ can not output a valid proof and therefore never wins game 5. Putting together equations 21-24 concludes our proof sketch of Theorem 5.3.

5.3.1 Instantiation based on SCasc

Towards an instantiation of our scheme, we need to choose a concrete family PS_k of NIWI proof systems and a family SIG_k of EUF-CMA signature schemes. We seek an interesting instantiation where reconfiguration of the PKI using a higher value of k (i.e., publishing a new CRS) leads to a system with increased security.

For this purpose, PS_k and SIG_k should be based on a family of assumptions that (presumably) become weaker as k grows such as the $\mathcal{D}_k$-MDDH assumption families from [11]. The k-SCasc assumption family seen in Section 2 is one interesting member of this class.

In the uniform adversary setting, [11, 16] shows that any $\mathcal{D}_k$-MDDH assumption family is enough to obtain a family of GS proof system $\mathsf{PS}_k := (\mathsf{CRSGen}_{\mathsf{PS}_k}, \mathsf{Prove}_{\mathsf{PS}_k}, \mathsf{Ver}_{\mathsf{PS}_k})$ with computationally indistinguishable CRS modes. More formally, one can show for any k that if $\mathcal{D}_k$-MDDH holds w.r.t. $\mathcal{G}$, then for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\text{ind-crs}}_{\mathsf{PS}_k,\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(CRS_{\mathsf{PS}_k}) = \mathtt{mode}] - \frac{1}{2}|$ is negligible in λ, where $CRS_{\mathsf{PS}_k} \leftarrow \mathsf{CRSGen}_{\mathsf{PS}_k}(\mathbb{G})$ and $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. If we base the construction in [11, 16] on the two-parameter variant of SCasc as defined in Section 4.2 (or of any other $\mathcal{D}_k$-MDDH assumption, which can be defined in a straightforward manner), we obtain a family of GS proof systems as required by our RSIG scheme.

Very recently, the concept of affine MACs was introduced in [5]. Basing their construction on the Naor-Reingold PRF, whose security follows from any $\mathcal{D}_k$-MDDH assumption, we can now construct a family of signature schemes SIG_k, where for each k we have that SIG_k is is EUF-CMA secure under $\mathcal{D}_k$-MDDH using the well-known fact that every PR-ID-CPA-secure IBE system implies an EUF-CMA-secure signature system.[7] Furthermore, we claim that using the same construction we can obtain a family of signature schemes as required by using the two-parameter variant of SCasc (or of any other $\mathcal{D}_k$-MDDH assumption) as the underlying assumption.

[7] In fact, [5] constructs an IB-KEM. It is straightforward to verify that a PR-IDKEM-CPA secure IB-KEM scheme also implies an EUF-CMA-secure signature scheme.

References

[1] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2), 6 (2012)

[2] Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of FOCS 1997. pp. 394–403. IEEE Computer Society (1997)

[3] Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. 61(3), 362–399 (2000)

[4] Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Proceedings of CRYPTO 1999. pp. 431–448. No. 1666 in Lecture Notes in Computer Science, Springer (1999)

[5] Blazy, O., Kiltz, E., Pan, J.: (hierarchical) identity-based encryption from affine message authentication. In: Proceedings of CRYPTO (1) 2014. pp. 408–425. No. 8616 in Lecture Notes in Computer Science, Springer (2014)

[6] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Proceedings of CRYPTO 2001. pp. 213–229. No. 2139 in Lecture Notes in Computer Science, Springer (2001)

[7] Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Proceedings of CRYPTO (1) 2014. pp. 480–499. No. 8616 in Lecture Notes in Computer Science, Springer (2014)

[8] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: Proceedings of FOCS 2010. pp. 501–510. IEEE Computer Society (2010)

[9] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. J. Cryptology 20(3), 265–294 (2007)

[10] Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography 2(2), 107–125 (1992)

[11] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for diffie-hellman assumptions. In: Proceedings of CRYPTO (2) 2013. pp. 129–147. No. 8043 in Lecture Notes in Computer Science, Springer (2013)

[12] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of FOCS 2013. pp. 40–49. IEEE Computer Society (2013)

[13] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of STOC 2008. pp. 197–206. ACM (2008)

[14] Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Proceedings of ASIACRYPT 2006. pp. 444–459. No. 4284 in Lecture Notes in Computer Science, Springer (2006)

[15] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Proceedings of EUROCRYPT 2008. pp. 415–432. No. 4965 in Lecture Notes in Computer Science, Springer (2008)

[16] Herold, G., Hesse, J., Hofheinz, D., Ràfols, C., Rupp, A.: Polynomial spaces: A new framework for composite-to-prime-order transformations. In: Proceedings of CRYPTO (1) 2014. pp. 261–279. No. 8616 in Lecture Notes in Computer Science, Springer (2014)

[17] Maurer, U.M., Yacobi, Y.: A non-interactive public-key distribution system. Des. Codes Cryptography 9(3), 305–316 (1996)

[18] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of STOC 2005. pp. 84–93. ACM (2005)

[19] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454 (2013), http://eprint.iacr.org/2013/454

[20] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of STOC 2014. pp. 475–484. ACM (2014)

[21] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Proceedings of CRYPTO 1984. pp. 47–53. No. 196 in Lecture Notes in Computer Science, Springer (1984)

[22] Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. IACR Cryptology ePrint Archive 2004, 332 (2004), http://eprint.iacr.org/2004/332

[23] Waters, B.: Efficient identity-based encryption without random oracles. In: Proceedings of EUROCRYPT 2005. pp. 114–127. No. 3494 in Lecture Notes in Computer Science, Springer (2005)