Policy-Based Signatures

Mihir Bellare* Georg Fuchsbauer†

Abstract

We introduce signatures where signers can only sign messages that conform to some policy, yet privacy of the policy is maintained. We provide definitions and show that policy-based signatures provide a framework which yields a unified view of many other existing types of signatures that now appear as special cases. We also show how still other primitives are easily realized using policy-based signatures as a building block. We provide generic constructions of policy-based signatures and then show how to achieve them efficiently.

Keywords: Signatures, policies, group signatures, NIZKs, Groth-Sahai proofs.

1 Introduction

We introduce *policy-based signatures (PBS)*, a (new) signing framework of both practical and theoretical interest. On the practical side, PBS offers flexible, fine-grained privacy-respecting authentication that is useful in a number of real-world applications. On the theoretical side, PBS unifies existing work, capturing other kinds of signatures as special cases or allowing them to be easily derived. We develop definitions, provide both generic and efficient constructions, and explore applications.

PBS IN BRIEF. In a standard digital signature scheme, a signer who has established a public verification key vk and a matching secret signing key sk can sign any message that it wants. Policy-based signatures offer more fine-grained authentication. A signer's secret key sk_p is associated to a policy p that allows the signer to produce a valid signature σ of a message m only if the message satisfies the policy, captured formally by requiring that (p, m) belongs to a *policy language* $L \subseteq \{0,1\}^* \times \{0,1\}^*$ associated to the scheme. This cannot be achieved if the signer creates her keys in a standalone way. In our model, a signer is issued a signing key sk_p for a particular policy p by an authority, as a function of a master secret key msk held by the authority. Verification that σ is a valid signature of m is then done with respect to the authority's public parameters pp. There are two security requirements. The first, unforgeability, says that producing a valid signature for message m is infeasible unless one has a secret key sk_p for some policy p such that $(p, m) \in L$. (You can only sign messages that you are allowed to sign.) The second, privacy, says that signatures do not reveal the policy under which they were created. (As part of this, we require signatures signed with the same key to be unlinkable.)

A trivial potential approach to achieving PBS is for the authority to certify the association of a signer's public key vk with its policy p via a signature of (p, vk) under msk, and have the signer include this certificate in its own signatures. This will provide unforgeability, but it will not provide privacy, because the policy must be revealed in the signature to allow for verification. Similarly, privacy in the absence of unforgeability is also trivial. The combination of the two requirements, however, results in a non-trivial goal.

PBS may be viewed as an authentication analogue of functional encryption [BSW11]. We may view the latter as allowing decryption to be policy-restricted rather than total, an authority issuing decryption

* Department of Computer Science and Engineering, University of California San Diego, La Jolla CA 92093. Supported in part by NSF grants CNS-1228890, CNS-1116800, CNS-0904380 and CCF-0915675.

† Institute of Science and Technology, Austria. Work done while at Bristol University, supported by EPSRC via grant EP/H043454/1.

keys in a way that enforces the policy. Correspondingly, in PBS, signing capability is policy-restricted rather than total, an authority issuing signing keys in a way that enforces the policy.

The setup of PBS is natural in a corporate or other hierarchical environment, where the manager wants to restrict the signing capability of different member entities based on their positions and privileges. However, the company policies underlying the restrictions need to be kept private.

We view PBS as a natural extension and generalization of a significant body of work on signatures that have privacy features, including group signatures [Cv91, BMW03], proxy signatures [MUO96], ring signatures [RST01, BKM06], mesh signatures [Boy07], anonymous proxy signatures [FP08], attribute-based signatures [MPR11] and anonymous credentials [CL01, BCKL08]. Our framework will allow us to unify or easily derive many previous signature variants.

Definitions. We allow the policy language to be any language in **NP**. This means that the policies that can be expressed and enforced are restricted neither in form nor in type, the only condition being that, given a witness, one can test in polynomial time whether a given policy allows a given message. A special case is languages in **P**. The latter already captures many typical applications, where one can directly test in polynomial time whether a given policy allows a given message.

We first provide a simple unforgeability definition and a natural, indistinguishability-based privacy definition. The latter says that the verifier cannot tell under which of two keys a signature was created assuming both policies associated to the keys permit the corresponding message. This already implies that the verifier cannot even decide whether two signatures were created using the same key. Yet we explain that applications call for even stronger notions. In response we provide simulatability and extractability notions. We show that their combination implies the simpler, more intuitive unforgeability and indistinguishability notions. Simulation+extractability emerges as a powerful security notion that enables a wide range of applications.

Constructions. Having created an ambitious target, namely PBS for languages in **NP** satisfying simulation+extractability, the first question that emerges is whether this can be achieved at all. We answer in the affirmative via two generic constructions based on standard primitives. The first uses ordinary signatures, IND-CPA encryption and standard non-interactive zero-knowledge (NIZK) proofs. The second uses only ordinary signatures and simulation-sound extractable NIZK proofs [Gro06].

While our generic constructions prove the theoretical feasibility of PBS, their use of general NIZKs makes them inefficient. We ask whether more efficient solutions may be given without using the random-oracle model [BR93]. We combine Groth-Sahai proofs [GS08] and structure-preserving signatures [AFG$^+$10] to design efficient PBS schemes for policy languages expressible via equations over a bilinear group. This construction requires a twist over usual applications of Groth-Sahai proofs, namely, in order to hide the policy, we swap the roles of constants and variables.

Applications. We illustrate applicability by showing how to derive a variety of other primitives from PBS. We start with showing that PBS implies attribute-based signatures, as modelled by [MPR11], in Section 5. In Section 7 we show that PBS even implies seemingly unrelated primitives like IND-CPA encryption and simulation-sound extractable NIZK proofs [Gro06]. By [Sah99] this means PBS implies IND-CCA encryption. We exploit this in Section 6 to show that PBS implies group signatures meeting the strong CCA version of the definition of [BMW03]. Finally, in Appendix E we show that PBS also implies signatures of knowledge [CL06]. These applications are illustrative rather than exhaustive, many more being possible.

Related work. In the world of digital signatures, extensions of functionality typically involve some form of delegation of signing rights: group signatures allow members to sign on behalf of a whole group, (anonymous) proxy signatures model delegation explicitly, and in attribute-based signatures and types of anonymous credentials, keys are also issued by an authority.

For most of these primitives, anonymity or privacy notions have been considered. A group signature, for example, should not reveal which group member produced a signature on behalf of the group (while an

authority can trace group signatures to their signer). In attribute-based signatures (ABS), users hold keys corresponding to their attributes and can sign messages with respect to a policy, which is a predicate over attributes. Users should only be able make signatures for policies that are satisfied by their attributes. Privacy for ABS means that a signature should reveal nothing about the attributes related to the key under which it was produced, other than the fact that it satisfies the policy.

In the models of primitives such as attribute-based signatures or mesh signatures, the policy itself is always public, as is the warrant, which specifies the policy in (even anonymous) proxy signatures. We ask whether this is a natural limitation of privacy notions, and whether it is inherently unavoidable that objects like the policy (which specify *why* the message could be signed) need to be public.

Consider the example of a company implementing a scheme where each employee gets a signing key and there is one public key which is used by outsiders to verify signatures in the name of the company. A group-signature scheme would allow every employee holding a key to sign on behalf of the company, but there is no fine-grained control over who is allowed to sign which documents. This can be achieved using attribute-based signatures, where each user is assigned attributes, and a message is signed with respect to a policy like `(`*CEO* `or (`*board member* `and` *general manager*`))`. However, it is questionable whether a verifier needs to know the company-internal policy used to sign a specific message, and there is no apparent reason he should know; all he needs to be assured of is that the message was signed by someone entitled to, but not who this person is, what she is entitled to sign, nor whether two messages were signed by the same person.

Another issue is that when using ABS we have to assume that the verifier can tell which messages can be signed under which policies. An attribute-based signature which is valid under the policy `(`*CEO* `or` *intern*`)` tells a verifier that it could have been produced by an intern, but it does not provide any guarantees as to whether an intern would have been entitled to sign the message. We ask whether it is possible to avoid having these types of public policies at all. PBS answers this in the affirmative.

The use of NIZKs for signatures begins with [BG90], who built an ordinary signature scheme from a NIZK, a PRF and a committment scheme. Encryption and ordinary signatures were combined with NIZKs to create group signatures in [BMW03]. Our first generic construction builds on these ideas. Our second generic construction, inspired by [BMT13], exploits the power of simulation-sound extractable NIZKs to give a conceptually simpler scheme that, in addition to the NIZK, uses only an ordinary signature scheme.

2 Preliminaries

Notations and conventions. The empty string is denoted by ε. If x is a (binary) string then $|x|$ is its length, $x[i]$ is its i-th bit and $x[i,j] = x[i] \dots x[j]$ for $1 \le i \le j \le |x|$. If S is a finite set then $|S|$ denotes its size and $s \leftarrow\!\!\$\, S$ denotes picking an element uniformly from S and assigning it to s. For $i \in \mathbb{N}$ we let $[i] = \{1, \dots, i\}$. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

Algorithms are randomized unless otherwise indicated. "PT" stands for "polynomial-time." By $y \leftarrow A(x_1, \dots; R)$, we denote the operation of running algorithm A on inputs $x_1, \dots$ and coins R and letting y denote the output. By $y \leftarrow\!\!\$\, A(x_1, \dots)$, we denote the operation of letting $y \leftarrow A(x_1, \dots; R)$ with R chosen at random. We denote by $[A(x_1, \dots)]$ the set of points that have positive probability of being output by A on inputs $x_1, \dots$. Adversaries are algorithms or tuples of algorithms. In the latter case, the running time of the adversary is the sum of the running times of all the algorithms in the tuple.

A map $\mathsf{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is said to be an **NP**-relation if it is computable in time polynomial in the length of its first input. For $x \in \{0,1\}^*$ we let $\mathsf{WS}_{\mathsf{R}}(x) = \{w : \mathsf{R}(x,w) = 1\}$ be the *witness set* of x. We let $\mathcal{L}(\mathsf{R}) = \{x : \mathsf{WS}_{\mathsf{R}}(x) \neq \emptyset\}$ be the *language* associated to R. The fact that R is an **NP**-relation means that $\mathcal{L}(\mathsf{R}) \in \mathbf{NP}$.

Game-playing framework. For our security definitions and proofs we use the code-based game-playing framework of [BR06]. A game **Exp** (Figure 1, for example) consists of a finite number of procedures. We

$\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{PBS}}$

proc INITIALIZE
$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$; $j \leftarrow 0$
Return pp

proc MAKESK(p)
$j \leftarrow j + 1$; $Q[j][1] \leftarrow p$
$Q[j][2] \leftarrow\!\!\$\ \mathsf{KeyGen}(pp, msk, p)$; $Q[j][3] \leftarrow \emptyset$

proc REVEALSK(i)
If $i \notin [1\,..\,j]$ then return $\perp$
$sk \leftarrow Q[i][2]$; $Q[i][2] \leftarrow \perp$; Return sk

proc SIGN(i, m, w)
If $i \notin [1\,..\,j]$ or $Q[i][2] = \perp$ then return $\perp$
$Q[i][3] \leftarrow Q[i][3] \cup \{m\}$
Return $\mathsf{Sign}(pp, Q[i][2], m, w)$

proc FINALIZE(m, σ)
If $\mathsf{Verify}(pp, m, \sigma) = 0$ then return false
For $i = 1, \ldots, j$ do
  If $(Q[i][1], m) \in \mathcal{L}(\mathsf{PC})$ then
    If $Q[i][2] = \perp$ or $m \in Q[i][3]$
      then return false
Return true

$\mathbf{Exp}^{\mathrm{IND}}_{\mathcal{PBS}}$

proc INITIALIZE
$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda)$; $b \leftarrow\!\!\$\ \{0,1\}$
Return (pp, msk)

proc LR(p_0, p_1, m, w_0, w_1)
If $\mathsf{PC}((p_0, m), w_0) = 0$ or $\mathsf{PC}((p_1, m), w_1) = 0$
  then return $\perp$
$sk_0 \leftarrow \mathsf{KeyGen}(msk, p_0)$
$sk_1 \leftarrow \mathsf{KeyGen}(msk, p_1)$
$\sigma_b \leftarrow \mathsf{Sign}(sk_b, m, w_b)$
Return (σ_b, sk_0, sk_1)

proc FINALIZE(b')
Return $(b = b')$

Figure 1: Games defining unforgeability and indistinguishability for PBS.

execute a game with an adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$ as follows. The adversary gets 1^λ as input. It can then query game procedures. Its first query must be to INITIALIZE with argument 1^λ, and its last to FINALIZE, and these must be the only queries to these oracles. In between it can query the other oracles as it wishes. The output of the execution, denoted $\mathbf{Exp}_{\mathcal{A}}(\lambda)$ is the output of FINALIZE. We denote by $\mathbf{Exp}_{\mathcal{A}}(\lambda) \Rightarrow y$ the event that this output is y. In code, boolean flags are assumed initialized to false, sets to $\emptyset$, integers to 0 and array entries to $\perp$. The running time of the adversary $\mathcal{A}$ is a function of λ in which oracle calls are assumed to take unit time.

3 Policy-Based Signatures

We now formally define policy-based signatures. A *policy checker* is an **NP**-relation $\mathsf{PC} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. The first input is a pair (p, m) representing a policy $p \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$, while the second input is a witness $w \in \{0,1\}^*$. The associated language $\mathcal{L}(\mathsf{PC}) = \{(p, m) : \mathsf{WS}_{\mathsf{PC}}((p, m)) \neq \emptyset\}$ is called the *policy language* associated to PC. That $(p, m) \in \mathcal{L}(\mathsf{PC})$ means that signing m is permitted under policy p. We say that (p, m, w) is PC-valid if $\mathsf{PC}((p, m), w) = 1$.

A *policy-based signature scheme* $\mathcal{PBS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a 4-tupe of PT algorithms:

(1) Setup : On input the unary-encoded security parameter 1^λ, setup algorithm Setup returns public parameters pp and a master secret key msk.

(2) KeyGen : On input msk, p, where $p \in \{0,1\}^*$ is a policy, key-generation algorithm KeyGen outputs a signing key sk for p.

(3) Sign : On input sk, m, w, where $m \in \{0,1\}^*$ is a message and $w \in \{0,1\}^*$ is a witness, signing algorithm Sign outputs a signature σ.

(4) Verify : On input pp, m, σ, verification algorithm Verify outputs a bit.

We say that the scheme is *correct* relative to policy checker PC if for all $\lambda \in \mathbb{N}$, all PC-valid (p, m, w), all $(pp, msk) \in [\mathsf{Setup}(1^\lambda)]$ and all $\sigma \in [\mathsf{Sign}(\mathsf{KeyGen}(msk, p), m, w)]$ we have $\mathsf{Verify}(pp, m, \sigma) = 1$.

UNFORGEABILITY. Our basic unforgeability requirement is that it is hard to create a valid signature of m without holding a key for some policy p such that $(p, m) \in \mathcal{L}(\mathsf{PC})$. The formalization is based on game $\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{PBS}}$ of Figure 1. For $\lambda \in \mathbb{N}$ we let $\mathbf{Adv}^{\mathrm{UF}}_{\mathcal{PBS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{PBS},\mathcal{A}} \Rightarrow \mathsf{true}]$. We say that $\mathcal{PBS}$ is *unforgeable*, or UF-*secure*, if $\mathbf{Adv}^{\mathrm{UF}}_{\mathcal{PBS},\mathcal{A}}(\cdot)$ is negligible for every PT $\mathcal{A}$. Via a MAKESK query, the adversary can have the game create a key for a policy p. Then, via SIGN, it can obtain a signature under this key for any message of its choice. (This models a chosen-message attack.) It may also, via its REVEALSK oracle, obtain the key itself. (This models corruption of users or the formation of collusions of users who pool their keys.) These queries naturally give the adversary the capability of creating signatures for certain messages, namely messages m such that for some p with $(p, m) \in \mathcal{L}(\mathsf{PC})$, it either obtained a key for p or obtained a signature for m. Unforgeability asks that it cannot sign any other messages. Note that we did not explicitly specify how Sign behaves when run on a key for p, and m, w with $\mathsf{PC}((p, m), w) = 0$. However, if it outputs a valid signature, this can be used to break UF-security.

INDISTINGUISHABILITY. Privacy for policy-based signatures requires that a signature not reveal the policy associated to the key and neither the witness that was used to create the signature. A first idea would be the following formalization: an adversary outputs a message m, two policies p_0, p_1, and two witnesses w_0, w_1, such that both (p_0, m, w_0) and (p_1, m, w_1) are PC-valid. For either p_0 or p_1 the experiment computes a secret key and uses it to produce a signature on m and gives it to the adversary, who wins if he can guess which policy was used. It turns out that this notion is too weak, as it does not guarantee that two signatures produced under the same secret key do not link, as seen as follows. Consider a scheme satisfying the security notion just sketched and modify it by attaching to each secret key a random string during key generation and alter Sign to append to the signature the random string contained in the secret key. Clearly, two signatures under the same key are linkable, but yet the scheme satisfies the definition. We therefore give the adversary the secret keys for both policies and a signature under one of them.

Let $\mathbf{Exp}^{\mathrm{IND}}_{\mathcal{PBS},\mathcal{A}}$ be the game defined on the right of Figure 1. we say that $\mathcal{PBS}$ has *indistinguishability* if for all PT adversaries $\mathcal{A}$ we have that $\mathbf{Adv}^{\mathrm{IND}}_{\mathcal{PBS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathrm{IND}}_{\mathcal{PBS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \frac{1}{2}$ is negligible in λ. We assume that either all policy descriptions p are of equal length, or we require that $\mathcal{A}$ output p_0 and p_1 with $|p_0| = |p_1|$.

Unlinkability could be formalized via a game where an adversary is given two signatures and must decide whether they were created using the same key. Indistinguishability implies unlinkability, as an adversary against the latter could be used to build another one against indistinguishability, who can simulate the unlinkability game by using the received signing keys to produce signatures.

3.1 Simulatability and Extractability

We now argue why the intuitively appealing notions of unforgeability and indistinguishability are unsatisfactory, in that they do not provide the level of security one hopes for. Moreover, it cannot be efficiently verified whether an adversary has won the unforgeability game, as this involves checking whether $(p, m) \in \mathcal{L}(\mathsf{PC})$ for all p queried to MAKESK and m from the adversary's final output. We thus replace unforgeability by an efficiently verifiable notion: *extractability* requires that from a valid signature, using a trapdoor one can extract a policy and a valid witness. To satisfy this notion, a signature must contain information on the policy and can thus not hide its length. For simplicity, we assume from now on that all policies are of the same length.

SIMULATABILITY. It turns out that even our strengthened notion of indistinguishability is too weak, in that it does not provide any security when the policy checker PC is such that for a message m there is only one p with $(p, m) \in \mathcal{L}(\mathsf{PC})$. (See our construction of group signatures in Section 6 for an example of such a PC.) To see why, consider a scheme which satisfies indistinguishability. Now modify the scheme so that the signing algorithm appends the policy to the signature. This scheme does evidently not hide the policy, yet still satisfies indistinguishability: In $\mathbf{Exp}^{\mathrm{IND}}_{\mathcal{PBS}}$, in order to satisfy $\mathsf{PC}((p_0, m), w_0) = 1 = \mathsf{PC}((p_1, m), w_1)$, the adversary must return $p_0 = p_1$. If the signatures in the original scheme have not revealed the bit b

$\mathbf{Exp}^{\text{SIM}}_{\mathcal{PBS}}$

proc INITIALIZE
$b \leftarrow\!\!\$\ \{0,1\}$; $j \leftarrow 0$
$(pp_0, msk_0, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
$(pp_1, msk_1) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
Return (pp_b, msk_b)

proc KEY(p)
$j \leftarrow j + 1$
$sk_0 \leftarrow\!\!\$\ \mathsf{SimKeyGen}(tr, p)$
$sk_1 \leftarrow\!\!\$\ \mathsf{KeyGen}(msk_1, p)$
$Q[j][1] \leftarrow p$; $Q[j][2] \leftarrow sk_1$
Return sk_b

proc SIGNATURE(i, m, w)
If $i \notin [1\,..\,j]$ then return $\bot$
If $\mathsf{PC}((Q[i][1], m), w) = 1$ then
  $\sigma_0 \leftarrow\!\!\$\ \mathsf{SimSign}(tr, m)$
Else $\sigma_0 \leftarrow \bot$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{Sign}(Q[i][2], m, w)$
Return σ_b

proc FINALIZE(b')
Return $(b = b')$

$\mathbf{Exp}^{\text{EXT}}_{\mathcal{PBS}}$

proc INITIALIZE
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
$Q_K \leftarrow \emptyset$; $Q_S \leftarrow \emptyset$
Return pp

proc SKEYGEN(p)
$sk \leftarrow\!\!\$\ \mathsf{SimKeyGen}(tr, p)$
$Q_K \leftarrow Q_K \cup \{p\}$
Return sk

proc SIMSIGN(m)
$\sigma \leftarrow\!\!\$\ \mathsf{SimSign}(tr, m)$
$Q_S \leftarrow Q_S \cup \{(m, \sigma)\}$
Return σ

proc FINALIZE(m, σ)
If $\mathsf{Verify}(pp, m, \sigma) = 0$ then return false
If $(m, \sigma) \in Q_S$ then return false
$(p, w) \leftarrow \mathsf{Extr}(tr, m, \sigma)$
If $p \notin Q_K$ or $\mathsf{PC}((p, m), w) = 0$
  then return true
Return false

Figure 2: Games defining simulatability and extractability for PBS

then attaching the same policy to both will not do so either. (Note however, that a scheme appending sk to a signature would not satisfy the notion.)

This issue can be overcome using a simulation-based definition: there is a simulator which can create simulated signatures without having access to any signing key or witness; and these signatures are indistinguishable from real signatures. We formalize *simulatability* by requiring that there exist the following algorithms: $\mathsf{SimSetup}$, which outputs parameters and a master key that are indistinguishable from those output by Setup and a trapdoor; $\mathsf{SKeyGen}$, which outputs keys indistinguishable from those output by KeyGen; and $\mathsf{SimSign}$, which on input the trapdoor and a message (but no signing key nor witness) produces signatures that are indistinguishable from regular signatures.

In particular, with $\mathbf{Exp}^{\text{SIM}}_{\mathcal{PBS}}$ defined on the left in Figure 2, we require that for every PT adversary $\mathcal{A}$ we have $\mathbf{Adv}^{\text{SIM}}_{\mathcal{PBS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\text{SIM}}_{\mathcal{PBS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \frac{1}{2}$ is negligible in λ. Note that in all our instantiations, tr will contain msk and $\mathsf{SKeyGen}$ will be defined as KeyGen. We included $\mathsf{SKeyGen}$ to make the definition more general.

EXTRACTABILITY. The notion of *soundness* for a NIZK proof system for an **NP**-relation R asks that it be impossible to produce a proof π which is valid for a statement x, although $x \notin \mathcal{L}(\mathsf{R})$. This has been strengthened to *simulation soundness* by Sahai [Sah99], stating that this even holds after the adversary has seen simulated proofs on statements of his choice. Since membership in $\mathcal{L}(\mathsf{R})$ may not be efficiently verifiable, it might not be efficiently decidable whether an adversary has broken soundness or not. The same holds for our notion of unforgeability, where FINALIZE checks whether $(Q[i][1], m) \in \mathcal{L}(\mathsf{PC})$. Although not a problem in itself, it can become one, for example when using the notion in a proof by game hopping, as a distinguisher between two games must efficiently determine whether an adversary has won the game. (See Appendix A for a proof that relies on this fact.)

For zero-knowledge proofs, the efficiently verifiable notion of *extractability* has been put forth [DMP88] to replace soundness: there exists an efficient extractor, which from a valid proof must extract a valid witness. If it fails to do so, the adversary wins. (Since there is no witness for an invalid statement, an adversary breaking soundness necessarily breaks extractability.) Extractability has been combined with

simulation soundness by Groth [Gro06]. We define our notion in the same spirit (which is also that of *"sim-ext" security* for signatures of knowledge [CL06]).

Let $\mathbf{Adv}^{\mathrm{EXT}}_{\mathcal{PBS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathrm{EXT}}_{\mathcal{PBS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}]$ with $\mathbf{Exp}^{\mathrm{EXT}}_{\mathcal{PBS}}$ defined on the right of Figure 2. We say that $\mathcal{PBS}$ has *extractability* if there exists an algorithm Extr, which taking a trapdoor, a message and a signature outputs a pair $(p, w) \in \{0,1\}^*$, such that $\mathbf{Adv}^{\mathrm{EXT}}_{\mathcal{PBS},\mathcal{A}}(\cdot)$ is negligible for every PT $\mathcal{A}$.

Although the definition might not seem completely intuitive at first, it implies that, as long as the adversary outputs a valid message/signature pair and does not simply copy a SimSign query/response pair, the only signed messages it can output are those that satisfy the policy of one of the queried keys: assume $\mathcal{A}$ outputs (m^*, σ^*) such that $(*)$ for all $p \in Q_K$: $(p, m^*) \notin \mathcal{L}(\mathsf{PC})$. Then let $(p^*, w^*) \leftarrow \mathsf{Extr}(tr, m, \sigma)$. If $\mathsf{PC}((p^*, m^*), w^*) = 0$, the adversary wins $\mathbf{Exp}^{\mathrm{EXT}}_{\mathcal{PBS}}$. On the other hand, if $\mathsf{PC}((p^*, m^*), w^*) = 1$ then $(p^*, m^*) \in \mathcal{L}(\mathsf{PC})$, thus by $(*)$ we have $p^* \notin Q_K$ and it wins too. Note that this notion corresponds to *strong unforgeability* for signature schemes.

Sim-Ext security implies indistinguishability and unforgeability. In Appendix A we show that our two latter security notions are indeed strengthenings of the former two:

Theorem 1. *Any policy-based signature scheme which satisfies simulatability satisfies indistinguishability. Any PBS scheme which satisfies simulatability and extractability satisfies unforgeability.*

4 Instantiations of Policy-Based Signatures

4.1 Generic Instantiation

We now show how to instantiate policy-based signatures satisfying simulatability and extractability (and, by Theorem 1, indistinguishability and unforgeability) for any **NP**-relation PC. A first approach could be the following, similar to the generic construction of group signatures in [BMW03]: The issuer creates a signature key pair (mvk, msk) and publishes mvk as pp; each user creates a key pair (vk_U, sk_U). When a user is issued a key for a policy p, the issuer signs $p \| vk_U$ and sends this certificate to the user. Now in order to sign a message m, the user first signs it using his personal key sk_U, establishing thus a chain $mvk \to vk_U \to m$ via the certificate and the signature. In order to remain anonymous, the actual signature is a (zero-knowledge) proof of knowledge of such a chain and the fact that the message satisfies the policy signed in the certificate.

While this approach would yield a scheme satisfying indistinguishability and unforgeability, it would fail to achieve extractability. We thus choose a different approach: The user's key is simply a signature from the issuer on the policy. Now to sign a message, the user first picks a key pair (ovk, osk) for a strongly unforgeable one-time signature scheme[1] and makes a zero-knowledge proof π that he knows either (I) an issuer signature on a policy p such that $(p, m) \in \mathcal{L}(\mathsf{PC})$ or (II) an issuer signature on ovk. Finally, he adds a signature on both the message and the proof of knowledge under ovk. As we will see, this construction satisfies both SIM (where the simulator can make a signature on ovk and use clause (II) for the proof) and EXT (as π is a proof of knowledge).

We formalize the above: Let $\mathcal{S}ig = (\mathsf{KeyGen}_{\mathrm{sig}}, \mathsf{Sign}_{\mathrm{sig}}, \mathsf{Verify}_{\mathrm{sig}})$ be a digital signature scheme, $\mathcal{OTS}ig = (\mathsf{KeyGen}_{\mathrm{ots}}, \mathsf{Sign}_{\mathrm{ots}}, \mathsf{Verify}_{\mathrm{ots}})$ a strongly unforgeable one-time signature scheme and let $\mathcal{PKE} = (\mathsf{KeyGen}_{\mathrm{pke}}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA-secure public-key encryption scheme. For a policy checker PC, we define the following **NP**-relation:

$$\begin{aligned}\big((pk, mvk, C_p, C_s, C_w, ovk, m), (p, s, w, \rho_p, \rho_s, \rho_w)\big) \;\in\; & R_{\mathrm{NP}} \\ \iff\; & C_p = \mathsf{Enc}(pk, p; \rho_p) \;\wedge\; C_s = \mathsf{Enc}(pk, s; \rho_s) \;\wedge\; C_w = \mathsf{Enc}(pk, w; \rho_w) \;\wedge \\ & \big[\big(\mathsf{Verify}_{\mathrm{sig}}(mvk, 1\|p, s) = 1 \;\wedge\; \mathsf{PC}((p, m), w) = 1\big) \vee \mathsf{Verify}_{\mathrm{sig}}(mvk, 0\|ovk, s) = 1\big]\end{aligned} \tag{1}$$

[1] In such a scheme it must be infeasible for an adversary, after receiving a verification key ovk and after obtaining a signature σ on one message m of his choice, to output a signature σ^* on a message m^*, such that $(m, \sigma) \neq (m^*, \sigma^*)$.

$\underline{\mathsf{Setup}(1^\lambda)}$
$crs \leftarrow\!\!\$\ \mathsf{Setup}_{\text{nizk}}(1^\lambda)$
$(pk, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pke}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{sig}}(1^\lambda)$
Return $pp \leftarrow (crs, pk, mvk)$ and msk

$\underline{\mathsf{KeyGen}(msk, p)}$
$s \leftarrow\!\!\$\ \mathsf{Sign}_{\text{sig}}(msk, 1\|p)$
Return $sk_p \leftarrow (pp, p, s)$

$\underline{\mathsf{Sign}(sk_p, m, w)}$
Parse $((crs, pk, mvk), p, s) \leftarrow sk_p$
If $\mathsf{PC}((p, m), w) = 0$ then return $\perp$
$(ovk, osk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{ots}}(1^\lambda)$
$\rho_p, \rho_s, \rho_w \leftarrow\!\!\$\ \{0,1\}^\lambda$; $C_p \leftarrow \mathsf{Enc}(pk, p; \rho_p)$
$C_s \leftarrow \mathsf{Enc}(pk, s; \rho_s)$; $C_w \leftarrow \mathsf{Enc}(pk, w; \rho_w)$
$\pi \leftarrow\!\!\$\ \mathsf{Prove}(crs, (pk, mvk, C_p, C_s, C_w, ovk, m), (p, s, w, \rho_p, \rho_s, \rho_w))$
$\tau \leftarrow\!\!\$\ \mathsf{Sign}_{\text{ots}}(osk, (m, C_p, C_s, C_w, \pi))$
Return $\sigma \leftarrow (ovk, C_p, C_s, C_w, \pi, \tau)$

$\underline{\mathsf{Verify}(pp, m, \sigma)}$
Parse $(crs, pk, mvk) \leftarrow pp$; $(ovk, C_p, C_s, C_w, \pi, \tau) \leftarrow \sigma$
Return 1 iff
$\mathsf{Verify}_{\text{nizk}}(crs, (pk, mvk, C_p, C_s, C_w, ovk, m), \pi) = 1$
and $\mathsf{Verify}_{\text{ots}}(ovk, (m, C_p, C_s, C_w, \pi), \tau) = 1$

$\underline{\mathsf{SimSetup}(1^\lambda)}$
$crs \leftarrow\!\!\$\ \mathsf{Setup}_{\text{nizk}}(1^\lambda)$
$(pk, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pke}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{sig}}(1^\lambda)$
Return $pp \leftarrow (crs, pk, mvk)$, msk and $tr \leftarrow (msk, dk)$

$\underline{\mathsf{SimKeyGen}((msk, dk), p)}$
$s \leftarrow\!\!\$\ \mathsf{Sign}_{\text{sig}}(msk, 1\|p)$
Return $sk_p \leftarrow (pp, p, s)$

$\underline{\mathsf{SimSign}((msk, dk), m)}$
$(ovk, osk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{ots}}(1^\lambda)$
$s \leftarrow\!\!\$\ \mathsf{Sign}_{\text{sig}}(msk, 0\|ovk)$
$\rho_p, \rho_s, \rho_w \leftarrow\!\!\$\ \{0,1\}^\lambda$; $C_p \leftarrow \mathsf{Enc}(pk, 0; \rho_p)$
$C_s \leftarrow \mathsf{Enc}(pk, s; \rho_s)$; $C_w \leftarrow \mathsf{Enc}(pk, 0; \rho_w)$
$\pi \leftarrow\!\!\$\ \mathsf{Prove}(crs, (pk, mvk, C_p, C_s, C_w, ovk, m), (0, s, 0, \rho_p, \rho_s, \rho_w))$
$\tau \leftarrow\!\!\$\ \mathsf{Sign}_{\text{ots}}(osk, (m, C_p, C_s, C_w, \pi))$
Return $\sigma \leftarrow (ovk, C_p, C_s, C_w, \pi, \tau)$

$\underline{\mathsf{Extr}((msk, dk), m, \sigma)}$
Parse $(ovk, C_p, C_s, C_w, \pi, \tau) \leftarrow \sigma$
$p \leftarrow \mathsf{Dec}(dk, C_p)$; $w \leftarrow \mathsf{Dec}(dk, C_w)$
Return (p, w)

Figure 3: Generic instantiation of PBS

Let $\mathcal{NIZK} = (\mathsf{Setup}_{\text{nizk}}, \mathsf{Prove}, \mathsf{Verify}_{\text{nizk}})$ be a non-interactive zero-knowledge (NIZK) proof system for $\mathcal{L}(R_{\text{NP}})$. Our instantiation $\mathcal{PBS}$ for a policy checker PC is detailed in Figure 3, and in Appendix B we prove the following:

Theorem 2. *If $\mathcal{PKE}$ satisfies IND-CPA, $\mathcal{S}ig$ is unforgeable under chosen-message attack, $\mathcal{O}t\mathcal{S}ig$ is a strongly unforgeable one-time signature scheme and $\mathcal{NIZK}$ is a NIZK proof system for $\mathcal{L}(R_{NP})$ then $\mathcal{PBS}[\mathcal{PKE}, \mathcal{S}ig, \mathcal{O}t\mathcal{S}ig, \mathcal{NIZK}]$, defined in Figure 3, satisfies simulatability and extractability.*

We now present a much simpler construction of PBS by relying on a more advanced cryptographic primitive: simulation-sound extractable (SSE) NIZK proofs [Gro06] (see Appendix E for the definition). Let $\mathcal{S}ig = (\mathsf{KeyGen}_{\text{sig}}, \mathsf{Sign}_{\text{sig}}, \mathsf{Verify}_{\text{sig}})$ be a signature scheme and for a policy checker PC let $\mathcal{NIZK} = (\mathsf{Setup}_{\text{zk}}, \mathsf{Prove}_{\text{zk}}, \mathsf{Verify}_{\text{zk}}, \mathsf{SimSetup}_{\text{zk}}, \mathsf{SimProve}_{\text{zk}}, \mathsf{Extr}_{\text{zk}})$ be an SSE-NIZK for the following **NP**-relation, whose statements are of the form $X = (vk, m)$ with witnesses $W = (p, c, w)$ and

$$((vk, m), (p, c, w)) \in R_{\text{NP}} \iff ((p, m), w) \in \mathsf{PC} \ \wedge \ \mathsf{Verify}_{\text{sig}}(vk, p, c) = 1$$

Then the scheme in Figure 4 is a PBS for PC which satisfies simulatability and extractability.

4.2 Efficient Instantiation via Groth-Sahai Proofs

Our efficient instantiation will be defined over a *bilinear group*. This is a tuple $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, G, H)$, where $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{T}$ are groups of prime order p, generated by G and H, respectively, and $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{T}$ is a bilinear map such that $e(G, H)$ generates $\mathbb{T}$. We denote the group operation multiplicatively and let $1_{\mathbb{G}}$ and $1_{\mathbb{H}}$ denote the neutral elements of $\mathbb{G}$ and $\mathbb{H}$. Groth-Sahai proofs [GS08] let us prove that there exists a set of elements $(X_1, \ldots, X_n, Y_1, \ldots, Y_\ell) \in \mathbb{G}^n \times \mathbb{H}^\ell$ which satisfy equations $\mathsf{E}(\underline{\vec{X}}, \underline{\vec{Y}})$ of the form

$\underline{\mathsf{Setup}(1^\lambda)}$
$crs \leftarrow\!\!\$\ \mathsf{Setup}_{\text{nizk}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{sig}}(1^\lambda)$
Return $pp \leftarrow (crs, mvk)$, msk

$\underline{\mathsf{KeyGen}(msk, p)}$
$c \leftarrow\!\!\$\ \mathsf{Sign}_{\text{sig}}(msk, p)$
Return $sk \leftarrow (pp, p, c)$

$\underline{\mathsf{Sign}(sk = ((crs, mvk), p, c), m, w)}$
Return $\sigma \leftarrow\!\!\$\ \mathsf{Prove}_{\text{nizk}}(crs, (mvk, m), (p, c, w))$

$\underline{\mathsf{Verify}(pp = (crs, mvk), m, \sigma)}$
Return $\mathsf{Verify}_{\text{nizk}}(crs, (mvk, m), \sigma)$

$\underline{\mathsf{SimSetup}(1^\lambda)}$
$(crs, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}_{\text{nizk}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{sig}}(1^\lambda)$
Return $pp \leftarrow (crs, mvk)$, msk, $tr_{\text{pbs}} \leftarrow (pp, msk, tr)$

$\underline{\mathsf{SimKeyGen}((pp, msk, tr), p)}$
$c \leftarrow\!\!\$\ \mathsf{Sign}_{\text{sig}}(msk, p)$
Return $sk \leftarrow (pp, p, c)$

$\underline{\mathsf{SimSign}(((crs, mvk), msk, tr), m)}$
Return $\sigma \leftarrow\!\!\$\ \mathsf{SimProve}_{\text{nizk}}(crs, tr, (mvk, m))$

$\underline{\mathsf{Extr}(((crs, mvk), msk, tr), m, \sigma)}$
$(p, c, w) \leftarrow \mathsf{Extr}_{\text{nizk}}(tr, (mvk, m), \sigma)$; Return (p, w)

Figure 4: PBS based on SSE NIZKs.

$$\prod_{i=1}^{k} e(P_i, Q_i) \prod_{j=1}^{\ell} e(A_j, \underline{Y_j}) \prod_{i=1}^{n} e(\underline{X_i}, B_i) \prod_{i=1}^{n} \prod_{j=1}^{\ell} e(\underline{X_i}, \underline{Y_j})^{\gamma_{ij}} = 1 \tag{2}$$

Such an equation E is called a *pairing-product equation*[2] (PPE) and is uniquely defined by its constants $\vec{P}, \vec{Q}, \vec{A}, \vec{B}$ and $\Gamma := (\gamma_{ij})_{i \in [n], j \in [\ell]}$. These equations have already found many uses in cryptography, of which the following two are relevant in our context: they can define the verification predicate of a digital signature (see [AFG+10] and the discussion therein), or witness the fact that a ciphertext encrypts a certain value (see Appendix F).

Groth and Sahai define a setup algorithm which on input a bilinear group outputs a common-reference string crs and an extraction key xk. On input crs, an equation E and a satisfying witness $(\vec{X}, \vec{Y})$, algorithm $\mathsf{Prove}_{\text{gs}}$ outputs a proof π. Proofs are verified by $\mathsf{Verify}_{\text{gs}}(crs, \mathsf{E}(\cdot, \cdot), \pi)$. Under the SXDH assumption (see [GS08] for the definition), Groth-Sahai proofs are *witness-indistinguishable* [FS90], that is, proofs for an equation using different witnesses are computationally indistinguishable. Moreover, they are *extractable*: From every valid proof π, $\mathsf{Extr}_{\text{gs}}(xk, \mathsf{E}(\cdot, \cdot), \pi)$ extracts a witness $(\vec{X}, \vec{Y})$ such that $\mathsf{E}(\vec{X}, \vec{Y}) = 1$, which means proofs are sound proofs of knowledge [DMP88].

When using Groth-Sahai proofs, we can define the following type of policy checker: the policy p defines an equation E_p as in (2) and $\mathsf{PC}((p, m), w) = 1$ iff $\mathsf{E}_p(m, w) = 1$, where $m \in \mathbb{G}^{n_m} \times \mathbb{H}^{\ell_m}$ and $w \in \mathbb{G}^{n_w} \times \mathbb{H}^{\ell_w}$. Now in order to hide the policy, we swap the roles of constants and variables, as this will enable us to hide the policy defined by the constants. Since Groth-Sahai proofs only allow us to prove knowledge of group elements, we first need to transform equations of the above form into a set of equivalent equations that do not contain exponents. To do so, we introduce auxiliary variables $\widehat{Y}_{ij}$, add $i \cdot j$ new equations and define the set $\mathsf{E}^{(\text{no-c})}$ as follows:

$$\prod e(P_i, Q_i) \prod e(A_j, \underline{Y_j}) \prod e(\underline{X_i}, B_i) \prod \prod e(\underline{X_i}, \underline{\widehat{Y}_{ij}}) = 1 \quad \wedge \quad \bigwedge_{i,j} e(G, \underline{\widehat{Y}_{ij}}) = e(G^{\gamma_{ij}}, \underline{Y_j}) \tag{3}$$

A witness $(\vec{X}, \vec{Y})$ satisfies E in (2) iff $(\vec{X}, \vec{Y}, (\widehat{Y}_{ij} := Y_j^{\gamma_{ij}})_{i,j})$ satisfies the set of equations $\mathsf{E}^{(\text{no-c})}$ in (3). Now we can swap the roles of constants and variables; that is, show that a (clear) message $(\vec{M}, \vec{N})$ satisfies a "hidden" policy defined by equation E, witnessed by elements $(\vec{V}, \vec{W})$, as we detail below.

Our second building block are structure-preserving signatures [AFG+10], which were designed to be combined with Groth-Sahai proofs: their keys, messages and signatures consist only of elements from $\mathbb{G}$ and $\mathbb{H}$ and signatures are verified by evaluating PPEs. Groth-Sahai proofs let us prove knowledge of keys, messages, and/or signatures such that these values satisfy signature verification, without revealing anything beyond this fact.

[2]This is a *simulatable* pairing-product equation, that is, one for which Groth-Sahai proofs can be made zero-knowledge.

Instantiation with Groth-Sahai proofs and structure-preserving signatures. To instantiate $\mathcal{PBS}$, we let $(\mathsf{Setup}_{\mathrm{gs}}, \mathsf{Prove}_{\mathrm{gs}}, \mathsf{Verify}_{\mathrm{gs}}, \mathsf{Extr}_{\mathrm{gs}})$ denote the Groth-Sahai proof system for PPEs, and let $(\mathsf{KeyGen}_{\mathrm{sp}}, \mathsf{Sign}_{\mathrm{sp}}, \mathsf{Verify}_{\mathrm{sp}})$ be a structure-preserving signature scheme that can sign vectors of group elements (e.g. one from [AFG+10]), for which we let $\mathsf{V}(vk, m, \sigma)$ denote the set of equations representing $\mathsf{Verify}_{\mathrm{sp}}(vk, m, \sigma)$. Moreover, we let $(\mathsf{KeyGen}_{\mathrm{ots}}, \mathsf{Sign}_{\mathrm{ots}}, \mathsf{Verify}_{\mathrm{ots}})$ be a strongly unforgeable one-time signature scheme whose verification keys are group elements (e.g. one from [AFG+10]). In the following we define a policy-based-signature scheme which follows closely the generic construction and whose security is proved analogously. Since Groth-Sahai proofs are proofs of knowledge, we do not need to encrypt the witnesses as in the generic construction.

One technically is that we need to express disjunctions as (the conjunction of) sets of equations. We do so by following Groth's approach in [Gro06]. We have to express the relation R_{NP} defined in Equation (1) in Section 4.1 as a set of PPEs. We start with showing how to express a disjunction. For an equation $\mathsf{E}(\vec{\underline{X}}, \vec{\underline{Y}})$ as defined in (2) above, we define the set $\mathsf{E}^{(\mathrm{sim})}(\vec{\underline{X}}, \vec{\underline{Y}})(\underline{T})$ having additional variables $Q'_1, \ldots Q'_k, T$ as

$$\prod e(P_i, \underline{Q'_i}) \prod e(A_j, \underline{Y_j}) \prod e(\underline{X_i}, B_i) \prod \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{ij}} = 1 \quad \wedge \quad \bigwedge_i \; e(\underline{T}, \underline{Q'_i} \cdot Q_i^{-1}) = 1 \tag{4}$$

When $T = 1_{\mathbb{G}}$ then setting $X_i \leftarrow 1_{\mathbb{G}}$ and $Q'_i, Y_j \leftarrow 1_{\mathbb{H}}$, for all i, j satisfies $\mathsf{E}^{(\mathrm{sim})}$. However, if $T \neq 1_{\mathbb{G}}$ then the only satisfying assignment to Q'_i is Q_i and thus $\vec{X}, \vec{Y}$ must satisfy the original equation E. In order to express a disjunction of two equations $\mathsf{E}_1(\vec{\underline{V}}, \vec{\underline{W}})$ and $\mathsf{E}_2(\vec{\underline{X}}, \vec{\underline{Y}})$, we convert them to $\mathsf{E}_1^{(\mathrm{sim})}(\vec{\underline{V}}, \vec{\underline{W}})(\underline{T_1})$ and $\mathsf{E}_2^{(\mathrm{sim})}(\vec{\underline{X}}, \vec{\underline{Y}})(\underline{T_2})$, respectively, and add the following equation E_T: $e(\underline{T_1} \cdot \underline{T_2} \cdot G^{-1}, H) = 1$. Now if we have a witness $(\vec{V}, \vec{W})$ for E_1, we can set $T_1 \leftarrow G$ and $T_2, \leftarrow 1_{\mathbb{G}}$ (and $\vec{X} \leftarrow \vec{1}_{\mathbb{G}}$ and $\vec{Y} \leftarrow \vec{1}_{\mathbb{H}}$); whereas if we have a witness for E_2, we set $T_1 \leftarrow 1_{\mathbb{G}}$ and $T_2 \leftarrow G$. In both cases we get a witness for the set $(\mathsf{E}_T, \mathsf{E}_1^{(\mathrm{sim})}, \mathsf{E}_2^{(\mathrm{sim})})$. However, E_T prevents us from setting both $\mathbb{T}_1 \leftarrow 1_{\mathbb{G}}$ and $\mathbb{T}_2 \leftarrow 1_{\mathbb{G}}$, which means we must have a witness for either E_1 or E_2. Note that this only works if the equations do not share any variables.

The transform $\mathsf{E}^{(\mathrm{sim})}$ of a set of equations $\mathsf{E}^{(\mathrm{no\text{-}c})}$ defined in (3) can be simplified by merely replacing the equations $e(G, \underline{\widehat{Y}_{ij}}) = e(G^{\gamma_{ij}}, \underline{Y_j})$ by $e(\underline{T}, \underline{\widehat{Y}_{ij}}) = e(G^{\gamma_{ij}}, \underline{Y_j})$. If $T = G$ then we are back in the situation of $\mathsf{E}^{(\mathrm{no\text{-}c})}$. However, if $T = 1_{\mathbb{G}}$ then setting all variables to $1_{\mathbb{G}}$ or $1_{\mathbb{H}}$, respectively, is a satisfying assignment for the transform.

Policy checkers for PPEs. Our policy checker PC for a policy $p = (\vec{P}, \vec{Q}, \vec{A}, \vec{B}, \vec{K}, \vec{L}, \Gamma = (\gamma_{ij}), \Delta = (\delta_{ij}), \Phi = (\phi_{ij}), \Psi = (\psi_{ij}))$, a message $m = (\vec{M}, \vec{N}) \in \mathbb{G}^{n_m} \times \mathbb{H}^{\ell_m}$ and a witness $w = (\vec{V}, \vec{W}) \in \mathbb{G}^{n_w} \times \mathbb{H}^{\ell_w}$ is defined as follows:

$$\begin{aligned} \mathsf{PC}((p, m), w) = 1 \iff & \prod e(P_i, Q_i) \prod e(A_j, N_j) \prod e(M_i, B_i) \prod e(K_j, W_j) \prod e(V_i, L_i) \\ & \prod \prod e(M_i, N_j)^{\gamma_{ij}} \prod \prod e(M_i, W_j)^{\delta_{ij}} \prod \prod e(V_i, N_j)^{\phi_{ij}} \prod \prod e(V_i, W_j)^{\psi_{ij}} = 1 \ , \end{aligned} \tag{5}$$

which is the most general form of a PPE over variables $\vec{M}, \vec{N}, \vec{V}$ and $\vec{W}$. Note that we assume that all policies are of a fixed length, since we cannot hide the *form* of the set of equations they define. Analogously to the transformation of $\mathsf{E} \rightarrow \mathsf{E}^{(\mathrm{no\text{-}c})}$ in (3), we express the exponents $\gamma_{ij}, \delta_{ij}, \phi_{ij}, \psi_{ij}$ (which are part of the policy) as group elements and make the equations simulatable by introducing variables T, $\widehat{N}_{ij}^{(1)}, \widehat{N}_{ij}^{(2)}, \widehat{W}_{ij}^{(1)}, \widehat{W}_{ij}^{(2)}$, and corresponding equations. We define:

$$\begin{aligned} \mathsf{PC}^{(\mathrm{sim})}((\underline{p}, m), \underline{w})(\underline{T}) = 1 \iff & \prod e(\underline{P_i}, \underline{Q_i}) \prod e(\underline{A_j}, N_j) \prod e(M_i, \underline{B_i}) \prod e(\underline{K_j}, \underline{W_j}) \prod e(\underline{V_i}, \underline{L_i}) \\ & \prod \prod e(M_i, \underline{N_{ij}^{(1)}}) \prod \prod e(M_i, \underline{W_{ij}^{(1)}}) \prod \prod e(\underline{V_i}, \underline{N_{ij}^{(2)}}) \prod \prod e(\underline{V_i}, \underline{W_{ij}^{(2)}}) = 1 \\ & \wedge \bigwedge_{i,j} \; e(\underline{T}, \underline{\widehat{N}_{ij}^{(1)}}) = e(\underline{G^{\gamma_{ij}}}, N_j) \; \wedge \; \bigwedge_{i,j} \; e(\underline{T}, \underline{\widehat{W}_{ij}^{(1)}}) = e(\underline{G^{\delta_{ij}}}, \underline{W_j}) \\ & \wedge \bigwedge_{i,j} \; e(\underline{T}, \underline{\widehat{N}_{ij}^{(2)}}) = e(\underline{G^{\phi_{ij}}}, N_j) \; \wedge \; \bigwedge_{i,j} \; e(\underline{T}, \underline{\widehat{W}_{ij}^{(2)}}) = e(\underline{G^{\psi_{ij}}}, \underline{W_j}) \end{aligned} \tag{6}$$

$\underline{\mathsf{Setup}(1^\lambda)}$
$(crs, xk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{gs}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{sp}}(1^\lambda)$
Return $pp \leftarrow (crs, mvk)$ and msk

$\underline{\mathsf{KeyGen}(msk, p)}$
$S \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{sp}}(msk, (G^1, p))$
Return $sk_p \leftarrow (pp, p, S)$

$\underline{\mathsf{Sign}(sk_p, m, w)}$
Parse $((crs, mvk), p, S) \leftarrow sk_p$
If $\mathsf{PC}((p, m), w) = 0$ then return $\bot$
$(ovk, osk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{ots}}(1^\lambda)$
$\pi \leftarrow\!\!\$\ \mathsf{Prove}_{\mathrm{gs}}(crs, \mathsf{E}^{(\mathrm{disj})}(mvk, \underline{p}, \underline{S}, \vec{\underline{1}}, \underline{w}, ovk, m, \underline{G}, \underline{1_{\mathbb{G}}}))$
$\tau \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{ots}}(osk, (m, \pi))$
Return $\sigma \leftarrow (ovk, \pi, \tau)$

$\underline{\mathsf{Verify}(pp, m, \sigma)}$
Parse $(crs, mvk) \leftarrow pp$; $(ovk, \pi, \tau) \leftarrow \sigma$
Return 1 iff
$\mathsf{Verify}_{\mathrm{gs}}(crs, (crs, \mathsf{E}^{(\mathrm{disj})}(mvk, \underline{\cdot}, \underline{\cdot}, \underline{\cdot}, \underline{\cdot}, ovk, m, \underline{\cdot}, \underline{\cdot})), \pi) = 1$
and $\mathsf{Verify}_{\mathrm{ots}}(ovk, (m, \pi), \tau) = 1$

$\underline{\mathsf{SimSetup}(1^\lambda)}$
$(crs, xk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{gs}}(1^\lambda)$
$(mvk, msk) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$
Return $pp \leftarrow (crs, mvk)$, msk, $tr \leftarrow (msk, xk)$

$\underline{\mathsf{SimKeyGen}((msk, xk), p)}$
$S \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{sp}}(msk, (G^1, p))$
Return $sk_p \leftarrow (pp, p, s)$

$\underline{\mathsf{SimSign}((msk, xk), m)}$
$(ovk, osk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{ots}}(1^\lambda)$
$S \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{sp}}(msk, (G^0, ovk))$
$\pi \leftarrow\!\!\$\ \mathsf{Prove}_{\mathrm{gs}}(crs, \mathsf{E}^{(\mathrm{disj})}(mvk, \vec{\underline{1}}, \vec{\underline{1}}, \underline{S}, \vec{\underline{1}}, ovk, m, \underline{1_{\mathbb{G}}}, \underline{G}))$
$\tau \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{ots}}(osk, (m, \pi))$
Return $\sigma \leftarrow (ovk, \pi, \tau)$

$\underline{\mathsf{Extr}((msk, xk), m, \sigma)}$
Parse $(ovk, \pi, \tau) \leftarrow \sigma$
$(p, S_1, S_2, w, T_1, T_2) \leftarrow \mathsf{Extr}_{\mathrm{gs}}(xk, \pi)$
Return (p, w)

Figure 5: Instantiation of PBS with Groth-Sahai proofs and structure-preserving signatures

If $T = G$ then $\widehat{N}_{ij}^{(1)} \leftarrow N_j^{\gamma_{ij}}$, $\widehat{N}_{ij}^{(2)} \leftarrow N_j^{\phi_{ij}}$, $\widehat{W}_{ij}^{(1)} \leftarrow W_j^{\delta_{ij}}$, $\widehat{W}_{ij}^{(2)} \leftarrow W_j^{\psi_{ij}}$ is the only satisfying assignment, and thus p and w need to satisfy the original equation. However, when $T = 1_{\mathbb{G}}$ then we can set all other variables to $1_{\mathbb{G}}$ or $1_{\mathbb{H}}$ as well.

Finally, in order to sign a policy (which contains the matrices $\Gamma, \Delta, \Phi, \Psi$ with entries in $\mathbb{Z}_p$), for any $\Xi = (\xi_{ij})_{ij}$, we define the projection onto $\mathbb{G}$ as $\Xi^{(\mathrm{prj})} := (G^{\xi_{ij}})_{ij}$. Henceforth, we assume that policies are given with their exponents projected to $\mathbb{G}$. We are now ready to express the generic equation in (1) as a set of pairing-product equations $\mathsf{E}^{(\mathrm{disj})}$ as follows. (Since the clauses of the disjunction must not have common variables, we use S_1 and S_2 for the signatures by the issuer.)

$$\begin{aligned}\mathsf{E}^{(\mathrm{disj})}(mvk, \underline{p}, \underline{S_1}, \underline{S_2}, \underline{w}, ovk, m, \underline{T_1}, \underline{T_2}) :\\ & e(\underline{T_1} \cdot \underline{T_2} \cdot G^{-1}, H) = 1\\ & \mathsf{V}^{(\mathrm{sim})}(mvk, (G^1, \vec{\underline{P}}, \vec{\underline{Q}}, \vec{\underline{A}}, \vec{\underline{B}}, \vec{\underline{K}}, \vec{\underline{L}}, \underline{\Gamma}, \underline{\Delta}, \underline{\Phi}, \underline{\Psi}), \underline{S_1})(\underline{T_1}) = 1\\ & \mathsf{PC}^{(\mathrm{sim})}\big((\vec{\underline{P}}, \vec{\underline{Q}}, \vec{\underline{A}}, \vec{\underline{B}}, \vec{\underline{K}}, \vec{\underline{L}}, \underline{\Gamma}, \underline{\Delta}, \underline{\Phi}, \underline{\Psi}), (\vec{M}, \vec{N}), (\vec{\underline{V}}, \vec{\underline{W}})\big)(\underline{T_1}) = 1\\ & \mathsf{V}^{(\mathrm{sim})}(mvk, (G^0, ovk), \underline{S_2})(\underline{T_2}) = 1\end{aligned}$$

Now that we have defined all the required concepts, our instantiation of PBS using Groth-Sahai proofs and structure-preserving signatures is quite straightforward. We present it in Figure 5.

Security is proven analogously to that of the scheme in Figure 3. Extractability follows from unforgeability of $\mathcal{S}ig_{\mathrm{sp}}$ and strong unforgeability of $\mathcal{O}t\mathcal{S}ig$, whereas simulatability follows from witness indistinguishability of Groth-Sahai proofs. Note that since we directly use a proof of knowledge, we need not simulate proofs as there are no ciphertexts, but instead simply change the witness used by Sign to the witness used by $\mathsf{SimSign}$.

A SIMPLE USE CASE. Messages that are elements of bilinear groups and policies demanding that they satisfy pairing-product equations will prove useful to construct other cryptographic schemes like group

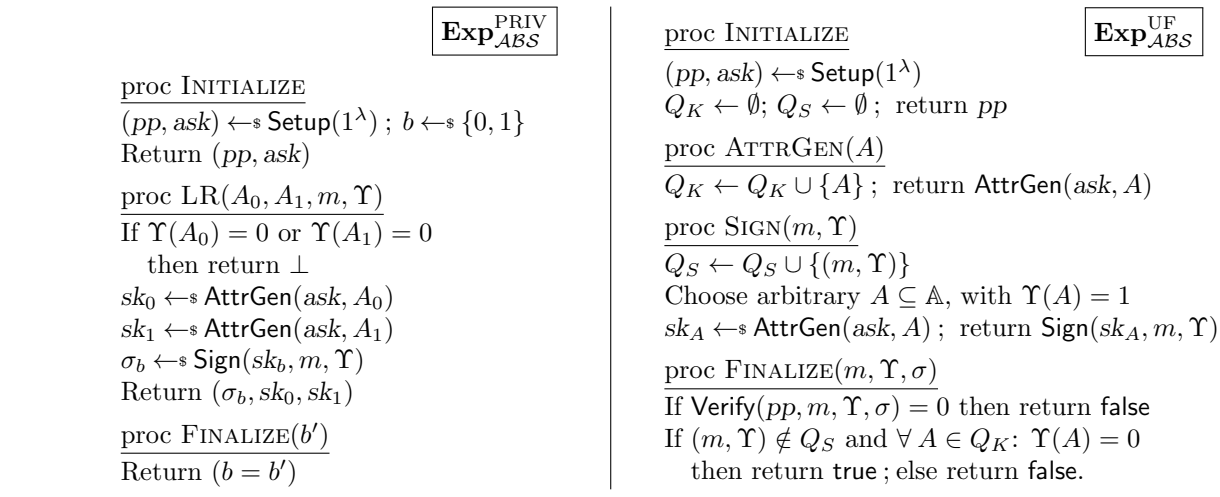

Figure 6: Games defining privacy and unforgeability for attribute-based signatures

signatures. However, our pairing-based instantiation might seem too abstract for deploying PBS to manage signing rights in a company—one of the motivations given in the introduction.

This is however not the case, as the following simple example shows: A company issues keys to their employees which should allow them to sign only messages $h\|m$ that start with a particular *header* h. This can be implemented by mapping messages $h\|m$ to $(F(h), F(m))$ via a collision-resistant hash function $F\colon \{0,1\}^* \to \mathbb{G}$. The policy p^* restricting signing to messages with a header h^* can then be expressed as $\mathsf{PC}((p^*, h\|m)) = 1 \Leftrightarrow e(F(h^*), H)\, e(F(h), H^{-1}) = 1$, which ensures $h = h^*$. Another possibility would be to additionally demand that an employee hold a credential, of which the required type could even depend on h^*, which she must use as a witness when signing. (There are numerous instantiations of credentials (digital signatures) which are verified via PPEs; see e.g. [AFG+10].)

5 Attribute-Based Signatures from Policy-Based Signatures

In attribute-based signatures (ABS) a trusted setup produces parameters and a master secret key. The latter is then used to issue keys for sets of attributes from a universe $\mathbb{A}$. Now a holder of such a key can sign messages w.r.t. to a predicate Υ over attributes, which must evaluate to 1 on the set of attributes for the key. The predicate is in the clear, so verification of a signature is w.r.t. the predicate.

The Model. In view of a generalization to multiple authorities, Maji et al. [MPR11] separate the setup algorithm into a trusted setup TSetup, which outputs public parameters tpk and ASetup, run by the attribute-issuing authority, which outputs a public/private key pair (apk, ask).

For the single-authority case (which we consider), TSetup and ASetup can be combined to Setup without weakening security: In the unforgeability game, both TSetup and ASetup are run by the experiment impersonating the attribute-issuing authority. Moreover, even though not explicitly stated, privacy also requires the pair $(apk, ask) \leftarrow\!\!\$\ \mathsf{ASetup}$ to be set up honestly.[3] For privacy, the adversary impersonates thus an "honest but curious" authority, which we model by giving the adversary the authority's secret key. Since security of our scheme is extraction-based, we cannot hope to achieve *perfect* privacy (meaning signatures produced with keys for different sets of attributes are distributed equally); we thus give a computational analog. We give a formal definition of the model.

We denote the message space by $\mathbb{M}$. A scheme $\mathcal{ABS}$ is parametrized by an *attribute universe* $\mathbb{A}$.

[3] If ask is maliciously set up so that $\mathsf{AttrGen}$ outputs a working key for one set of attributes and a key leading to invalid signatures for another set then privacy does not hold.

A *claim predicate* over $\mathbb{A}$ is a monotone boolean function $\Upsilon\colon \mathcal{P}(\mathbb{A}) \to \{0,1\}$. On input the security parameter 1^λ, Setup outputs public parameters pp and the authority's secret key ask. On input ask and $A \subseteq \mathbb{A}$, $\mathsf{AttrGen}$ outputs a secret key sk_A. On input sk_A, m, Υ, where $\Upsilon(A) = 1$, Sign outputs a signature σ. On input pp, m, Υ, σ, Verify outputs a bit. Correctness requires that for all $\lambda \in \mathbb{N}$, all $(pp, ask) \in [\mathsf{Setup}(1^\lambda)]$, all $m \in \mathbb{M}$, all $A \subseteq \mathbb{A}$, all Υ with $\Upsilon(A) = 1$ and all $\sigma \in [\mathsf{Sign}(\mathsf{AttrGen}(ask, A), m, \Upsilon)]$, we have $\mathsf{Verify}(pp, m, \Upsilon, \sigma) = 1$.

We say that $\mathcal{ABS}$ has *privacy* if $\mathbf{Adv}^{\mathrm{PRIV}}_{\mathcal{ABS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathrm{PRIV}}_{\mathcal{ABS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \frac{1}{2}$ is negligible in λ for all PT $\mathcal{A}$, with $\mathbf{Exp}^{\mathrm{PRIV}}_{\mathcal{ABS},\mathcal{A}}$ defined on the left of Figure 6. On the right we define the game $\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{ABS},\mathcal{A}}$ and say that $\mathcal{ABS}$ is *unforgeable* if $\mathbf{Adv}^{\mathrm{UF}}_{\mathcal{ABS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{ABS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}]$ is negligible in λ for all PT $\mathcal{A}$. Although our notion of privacy is only computational, it is stronger in another aspect: the adversary gets the two signing keys.[4]

Construction. Let $\mathbb{Y}$ denote the set of all monotone boolean functions over $\mathbb{A}$. We define a policy checker PC for our policy-based signature $\mathcal{PBS}$ which instantiates ABS. A policy is a set $A \subseteq \mathbb{A}$ and a message for $\mathcal{PBS}$ is in $\mathbb{Y} \times \mathbb{M}$, i.e., a claim predicate and the actual message. A $\mathcal{PBS}$ message satisfies a policy if the set of attributes defining the policy satisfy the predicate contained in the message. The policy checker PC for $\mathcal{PBS}$ is efficiently decidable (thus no witnesses are required) and is defined as:

$$\mathsf{PC}\colon \mathbb{A} \times (\mathbb{Y} \times \mathbb{M}) \to \{0,1\} \qquad (A, (\Upsilon, m)) \mapsto \Upsilon(A)$$

Let $\mathcal{PBS} = (\mathsf{Setup}_{\mathrm{pbs}}, \mathsf{KeyGen}_{\mathrm{pbs}}, \mathsf{Sign}_{\mathrm{pbs}}, \mathsf{Verify}_{\mathrm{pbs}})$ be a policy-based signature scheme for PC. We define $\mathcal{ABS}[\mathcal{PBS}] = (\mathsf{Setup}, \mathsf{AttrGen}, \mathsf{Sign}, \mathsf{Verify})$ as follows:

$\underline{\mathsf{Setup}(1^\lambda)}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{pbs}}(1^\lambda)$; return pp, $ask \leftarrow msk$

$\underline{\mathsf{AttrGen}(ask, A)}$
Return $sk_A \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{pbs}}(ask, A)$

$\underline{\mathsf{Sign}(pp, sk_A, m, \Upsilon)}$
Return $\mathsf{Sign}_{\mathrm{pbs}}(sk_A, (\Upsilon, m))$

$\underline{\mathsf{Verify}(pp, m, \Upsilon, \sigma)}$
Return $\mathsf{Verify}_{\mathrm{pbs}}(pp, (\Upsilon, m), \sigma)$

Theorem 3. *If $\mathcal{PBS}$ is a policy-based signature scheme satisfying indistinguishability and unforgeability then $\mathcal{ABS}[\mathcal{PBS}]$ is an attribute-based signature scheme satisfying privacy and unforgeability.*

The proof can be found in Appendix C. Using the strategy to express disjunctions of statements as sets of pairing-product equations outlined in Section 4.2, we can express PC as a set of PPEs and thus use our efficient PBS-implementation to construct an efficient ABS. We conclude by remarking that we could also instantiate *key-policy* ABS, where the key is associated with a predicate and the message with a set of attributes. We would simply define $\mathsf{PC}\colon \mathbb{Y} \times (\mathbb{A} \times \mathbb{M}) \to \{0,1\}$, $\mathsf{PC}(\Upsilon, (A, m)) = \Upsilon(A)$.

6 CCA-Secure Group Signatures from Policy-Based Signatures

The BMW Model. As defined in [BMW03], a *group signature scheme* $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ is a 4-tuple of PT algorithms. On input the security parameter 1^λ and the group size 1^n, group-key-generation algorithm GKg returns the group public key gpk, the manager's secret key $gmsk$ and a vector of member secret keys $\boldsymbol{gsk}$. On input $\boldsymbol{gsk}[i]$ and a message $m \in \{0,1\}^*$, group signing algorithm GSig returns a group signature γ by member i on m. On input gpk, m, γ, verification algorithm GVf outputs a bit. On input $gmsk, m, \gamma$, the opening algorithm Open returns an identity $i \in [n]$ or $\perp$.

We say that $\mathcal{GS}$ is correct if for all $\lambda, n \in \mathbb{N}$, all $(gpk, gmsk, \boldsymbol{gsk}) \leftarrow\!\!\$\ \mathsf{GKg}(1^\lambda, 1^n)$, all $1 \le i \le n$, and all $m \in \{0,1\}^*$, we have $\mathsf{GVf}(gpk, m, \mathsf{GSig}(\boldsymbol{gsk}[i], m)) = 1$ and $\mathsf{Open}(gmsk, m, \mathsf{GSig}(\boldsymbol{gsk}[i], m)) = i$.

Security for $\mathcal{GS}$ is defined via the experiments $\mathbf{Exp}^{\mathrm{ANON}}_{\mathcal{GS}}$ and $\mathbf{Exp}^{\mathrm{TRC}}_{\mathcal{GS}}$ defined in Figure 7. Following [BMW03], in $\mathbf{Exp}^{\mathrm{ANON}}_{\mathcal{GS}}$ we allow the adversary only one call to his LR oracle. We say that $\mathcal{GS}$ is *fully*

[4] For group signatures, this notion was termed *full* anonymity [BMW03], as opposed to *selfless* anonymity [BS04], where users are able to recognize signatures produced with their own signing key.

$\mathbf{Exp}^{\text{ANON}}_{\mathcal{GS}}$

proc INITIALIZE
$(gpk, gmsk, \boldsymbol{gsk}) \leftarrow\!\!\$\ \mathsf{GKg}(1^k, 1^n)$
$b \leftarrow\!\!\$\ \{0,1\}$; $Q \leftarrow \emptyset$
Return $(gpk, \boldsymbol{gsk})$

proc LR(i_0, i_1, m)
$\gamma \leftarrow\!\!\$\ \mathsf{GSig}(\boldsymbol{gsk}[i_b], m)$
$Q \leftarrow Q \cup \{\gamma\}$; return γ

proc OPEN(m, γ)
If $\gamma \in Q$ then return $\bot$
Return $\mathsf{Open}(gmsk, m, \gamma)$

proc FINALIZE(b')
Return $(b = b')$

$\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$

proc INITIALIZE
$(gpk, gmsk, \boldsymbol{gsk}) \leftarrow\!\!\$\ \mathsf{GKg}(1^k, 1^n)$
$Q_C \leftarrow \emptyset$; $Q_S \leftarrow \emptyset$; return $(gpk, gmsk)$

proc CORRUPT(i)
$Q_C \leftarrow Q_C \cup \{i\}$; return $\boldsymbol{gsk}[i]$

proc GSIG(i, m)
$Q_S \leftarrow Q_S \cup \{(i, m)\}$; return $\mathsf{GSig}(\boldsymbol{gsk}[i], m)$

proc FINALIZE(m, γ)
If $\mathsf{GVf}(gpk, m, \gamma) = 0$ then return false
If $\mathsf{Open}(gmsk, m, \gamma) = \bot$ then return true
$i \leftarrow \mathsf{Open}(gmsk, m, \gamma)$
If $i \in [n]$, $i \notin Q_C$ and $(i, m) \notin Q_S$ then return true
Return false

Figure 7: Games defining full anonymity and traceability for group signatures

anonymous if for all PT adversaries $\mathcal{A}$ the following is negligible in λ: $\mathbf{Adv}^{\text{ANON}}_{\mathcal{GS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{ANON}}_{\mathcal{GS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \frac{1}{2}$. We say that $\mathcal{GS}$ is *traceable* if for all PT adversaries $\mathcal{A}$ the following is negligible in λ: $\mathbf{Adv}^{\text{TRC}}_{\mathcal{GS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}]$.

CONSTRUCTION. We show how to construct group signatures (GS) from a CCA-secure public key encryption scheme and policy-based signatures. Since the former can be constructed from the latter (as we show in Appendix E), this means that PBS imply GS. The main idea is to define a group signature as a ciphertext plus a PBS. When making a group signature on a message m, a member is supposed to encrypt her identity as c and then sign (c, m). The policy for which the member gets a PBS-key ensures that c must be an encryption of the member's identity.

Let $\mathcal{PKE} = (\mathsf{KeyGen}_{\text{pke}}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme satisfying IND-CCA and let $\mathcal{PBS} = (\mathsf{Setup}, \mathsf{KeyGen}_{\text{pbs}}, \mathsf{Sign}, \mathsf{Verify})$ be a PBS for the following **NP**-relation:

$$\mathsf{PC}\big(((ek, i), (c, m)), r\big) \iff c = \mathsf{Enc}(ek, i; r) \ . \tag{7}$$

Our group-signature scheme $\mathcal{GS}[\mathcal{PKE}, \mathcal{PBS}]$ is defined as follows:

$\underline{\mathsf{GKg}(1^\lambda, 1^n)}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pke}}(1^\lambda)$
For $i = 1, \ldots, n$ do
$sk_i \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pbs}}(msk, (ek, i))$
$\boldsymbol{gsk}[i] \leftarrow (pp, ek, i, sk_i)$
Return $(gpk \leftarrow (pp, ek), gmsk \leftarrow dk, \boldsymbol{gsk})$

$\underline{\mathsf{GSig}((pp, ek, i, sk_i), m)}$
$r \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c \leftarrow \mathsf{Enc}(ek, i; r)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(sk_i, (c, m), r)$
Return (c, σ)

$\underline{\mathsf{GVf}((pp, ek), m, (c, \sigma))}$
Return $\mathsf{Verify}(pp, (c, m), \sigma)$

$\underline{\mathsf{Open}(gmsk, m, (c, \sigma))}$
If $\mathsf{Verify}(pp, (c, m), \sigma) = 0$
Then return $\bot$
Return $\mathsf{Dec}(gmsk, c)$

Theorem 4. *If $\mathcal{PBS}$ is a policy-based signature scheme satisfying simulatability and extractability and $\mathcal{PKE}$ is a public-key encryption scheme satisfying IND-CCA then $\mathcal{GS}[\mathcal{PKE}, \mathcal{PBS}]$ is a group-signature scheme satisfying full anonymity and traceability.*

The proof can be found in Appendix D. In Appendix F we give an encryption scheme such that (7) lies in the language of our efficient PBS from Section 4.2.

7 Other Primitives Implied by PBS

Simulation-sound extractable NIZK proofs. Groth [Gro06] introduced a notion of simulation-soundness for NIZK proofs of knowledge. It requires that even when an adversary is provided with an oracle for simulated proofs, it cannot produce a new valid proof from which the extractor fails to extract a witness. We refer to Appendix E for the definition and more details.

Let R be an **NP**-relation and let PC and p^* be such that $\mathsf{PC}((p^*, x), w) = \mathsf{R}(x, w)$. Let $\mathcal{PBS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SimSetup}, \mathsf{SKeyGen}, \mathsf{SimSign}, \mathsf{Extr})$ be a PBS for PC which satisfies simulatability and extractability. Then the following is a simulation-sound extractable NIZK proof system:

$\underline{\mathsf{Setup}_{\mathrm{zk}}(1^\lambda)}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$sk \leftarrow \mathsf{KeyGen}(msk, p^*)$
Return $crs \leftarrow (pp, sk)$

$\underline{\mathsf{Prove}((pp, sk), x, w)}$
Return $\pi \leftarrow \mathsf{Sign}(sk, x, w)$

$\underline{\mathsf{Verify}_{\mathrm{zk}}((pp, sk), x, \pi)}$
Return $\mathsf{Verify}(pp, x, \pi)$

$\underline{\mathsf{Extr}_{\mathrm{zk}}(tr, x, \pi)}$
$(p, w) \leftarrow \mathsf{Extr}(tr, x, \pi)$
Return w

$\underline{\mathsf{SimSetup}_{\mathrm{zk}}(1^\lambda)}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
$sk \leftarrow \mathsf{SKeyGen}(tr, p^*)$
Return $crs \leftarrow (pp, sk)$ and tr

$\underline{\mathsf{SimProve}((pp, sk), tr, x)}$
Return $\pi \leftarrow \mathsf{SimSign}(tr, x)$

Public-key encryption. Interpreting policies as plaintexts hidden in a signature, which can be "decrypted" using Extr, PBS even imply public-key encryption: Let $\mathcal{PBS}$ be for PC s.t. $\mathsf{PC}((p, m^*), w^*) = 1$ for some m^*, w^* and all p. Then the following as an IND-CPA secure PKE:

$\underline{\mathsf{KeyGen}_{\mathrm{pke}}(1^\lambda)}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
Return $(pk \leftarrow (pp, msk),\ dk \leftarrow tr)$

$\underline{\mathsf{Enc}((pp, msk), x)}$
$sk \leftarrow\!\!\$\ \mathsf{KeyGen}(msk, x)$
Return $c \leftarrow\!\!\$\ \mathsf{Sign}(sk, m^*, w^*)$

$\underline{\mathsf{Dec}(dk, c)}$
$(x, w) \leftarrow \mathsf{Extr}(dk, m^*, c)$
Return x

Using the results by Sahai [Sah99], the above NIZK and PKE can be combined to construct a CCA-secure PKE from PBS. We refer to Appendix E for the details, where we also construct signatures of knowledge [CL06] from PBS.

References

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, August 2010.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, March 2008.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, August 1990.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, March 2006.

[BMT13] Mihir Bellare, Sarah Meiklejohn, and Susan Thomson. Key-Versatile Signatures and Applications: RKA, KDM and Joint Enc/Sig. Cryptology ePrint Archive, Report 2013/326, 2013.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, May 2003.

[Boy07] Xavier Boyen. Mesh signatures. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 210–227. Springer, May 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 168–177. ACM Press, October 2004.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, March 2011.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, August 2006.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, April 1991.

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 52–72. Springer, August 1988.

[FP08] Georg Fuchsbauer and David Pointcheval. Anonymous proxy signatures. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 201–217. Springer, September 2008.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd STOC*, pages 416–426. ACM, 1990.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, December 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer, February 2011.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *ACM CCS 96*, pages 48–57. ACM Press, March 1996.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd STOC*, pages 427–437. ACM, 1990.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, December 2001.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

proc INITIALIZE // $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$j \leftarrow 0\ ;\ Q_S \leftarrow \emptyset$
Return pp

proc MAKESK(p) // $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$
$j \leftarrow j+1\ ;\ Q[j][1] \leftarrow p$
$Q[j][2] \leftarrow\!\!\$\ \mathsf{KeyGen}(pp, msk, p)\ ;\ Q[j][3] \leftarrow \emptyset$

proc REVEALSK(i) // $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$
If $i \notin [1\,..\,j]$ then return $\bot$
$sk \leftarrow Q[i][2]\ ;\ Q[i][2] \leftarrow \bot\ ;$ return sk

proc SIGN(i, m, w) // $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$
If $i \notin [1\,..\,j]$ then return $\bot$
If $Q[i][2] = \bot$ then return $\bot$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(pp, Q[i][2], m, w)$
$Q_S \leftarrow Q_S \cup \{(m, \sigma)\}$; return σ

proc FINALIZE(m, σ) // $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$
If $\mathsf{Verify}(pp, m, \sigma) = 0$ then return false
$(p, w) \leftarrow \mathsf{Extr}(tr, m, \sigma)$
If $(m, \sigma) \in Q_S$ then return false
For $i = 1, \ldots, j$ do
  If $Q[i][1] = p$ and $Q[i][2] = \bot$
    If $\mathsf{PC}((p, m), w) = 1$ then return false
Return true

proc INITIALIZE // $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
$j \leftarrow 0\ ;\ Q_S \leftarrow \emptyset$
Return pp

proc MAKESK(p) // $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$
$j \leftarrow j+1\ ;\ Q[j][1] \leftarrow p$
$Q[j][2] \leftarrow\!\!\$\ \mathsf{SKeyGen}(pp, tr, p)\ ;\ Q[j][3] \leftarrow \emptyset$

proc REVEALSK(i) // $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$
If $i \notin [1\,..\,j]$ then return $\bot$
$sk \leftarrow Q[i][2]\ ;\ Q[i][2] \leftarrow \bot\ ;$ return sk

proc SIGN(i, m, w) // $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$
If $i \notin [1\,..\,j]$ then return $\bot$
If $Q[i][2] = \bot$ then return $\bot$
If $\mathsf{PC}((Q[i][1], m), w) = 0$ then return $\bot$
$\sigma \leftarrow\!\!\$\ \mathsf{SimSign}(pp, tr, m)$
$Q_S \leftarrow Q_S \cup \{(m, \sigma)\}$; return σ

proc FINALIZE(m, σ) // $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$
If $\mathsf{Verify}(pp, m, \sigma) = 0$ then return false
$(p, w) \leftarrow \mathsf{Extr}(tr, m, \sigma)$
If $(m, \sigma) \in Q_S$ then return false
For $i = 1, \ldots, j$ do
  If $Q[i][1] = p$ and $Q[i][2] = \bot$
    If $\mathsf{PC}((p, m), w) = 1$ then return false
Return true

Figure 8: Games $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$ and $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$ in the proof SIM/EXT ⇒ UF

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, May 2005.

A Proofs that SIM and EXT Imply IND and UF

SIMULATABILITY IMPLIES INDISTINGUISHABILITY. Assuming an adversary $\mathcal{A}$ against indistinguishability, we construct an adversary $\mathcal{B}$ against the simulatability:

$\mathcal{B}$ receives (pp, msk), sets $j \leftarrow 1$, chooses $d \leftarrow\!\!\$\ \{0,1\}$ and runs $\mathcal{A}$ on (pp, msk). Whenever $\mathcal{A}$ queries $\mathsf{LR}(p_0, p_1, m, w_0, w_1)$, if $\mathsf{PC}((p_0, m), w_0) = 0$ or $\mathsf{PC}((p_1, m), w_1) = 0$, it returns $\bot$; otherwise it queries $sk_0 \leftarrow\!\!\$\ \mathsf{Key}(p_0)$ and $sk_1 \leftarrow\!\!\$\ \text{KEY}(p_1)$ and $\sigma_d \leftarrow\!\!\$\ \text{SIGNATURE}(j+d, m, w_d)$ and sets $j \leftarrow j+2$; it returns (σ_d, sk_0, sk_1) to $\mathcal{A}$. When $\mathcal{A}$ terminates outputting b', $\mathcal{B}$ outputs 1 if $(b' = d)$ and 0 otherwise.

If in $\mathbf{Exp}^{\text{SIM}}_{\mathcal{PBS},\mathcal{B}}$, the challenger's bit is 1 then $\mathcal{B}$ perfectly simulates $\mathbf{Exp}^{\text{IND}}_{\mathcal{PBS}}$ for $\mathcal{A}$; if on the other hand the bit is 0 then the bit d chosen by $\mathcal{B}$ is perfectly hidden from $\mathcal{A}$, meaning $\mathcal{B}$ outputs 1 with probability $\frac{1}{2}$. Together, this yields $\mathbf{Adv}^{\text{SIM}}_{\mathcal{PBS},\mathcal{B}} = \frac{1}{2} \cdot \mathbf{Adv}^{\text{IND}}_{\mathcal{PBS},\mathcal{A}}$.

SIMULATABILITY AND EXTRACTABILITY IMPLY UNFORGEABILITY. The benefit of defining unforgeability with the help of an extractor is that the experiment is efficiently decidable, as there are no more conditions like $(p, m) \in \mathcal{L}(\mathsf{PC})$. In efficiently decidable experiments, we can replace real signatures and keys by simulated ones without changing the adversary's behavior, since we could build a distinguisher that breaks the SIM.

Recall $\mathbf{Exp}^{\text{IND}}_{\mathcal{PBS}}$, defined in Figure 1. Now consider the modification $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$ given in Figure 8, whose FINALIZE procedure is basically that of $\mathbf{Exp}^{\text{EXT}}_{\mathcal{PBS}}$: the signature must be valid and (m, σ) must not

be a response of the SIGN oracle. We then apply the extractor of $\mathcal{PBS}$ to get (p, w) and the adversary wins if either no secret key for p was ever revealed or if $\mathsf{PC}((p, m), w) = 0$.

Claim. UF-(1) $\Rightarrow$ UF.

We show how to use an adversary $\mathcal{A}$ winning UF to construct $\mathcal{B}$ which wins UF-(1). What makes the proof not quite straightforward is the fact that $\mathcal{A}$ might win UF by outputting a pair (m, σ) which is a query and response of the SIGN oracle,[5] whereas this would make UF-(1) immediately return false, since $(m, \sigma) \in Q_S$.

We start with the simpler case, where $\mathcal{A}$ never outputs a query/response pair from the SIGN oracle. Our reduction $\mathcal{B}$ simply forwards all messages between its $\mathbf{Exp}^{\text{UF-(1)}}_{\mathcal{PBS}}$-challenger and $\mathcal{A}$. If $\mathcal{A}$ wins UF by outputting (m^*, σ^*) then we have (1) $\mathsf{Verify}(pp, m^*, \sigma^*) = 1$ and (2) $(m^*, \sigma^*) \notin Q_S$ (by assumption). Let $(\bar{p}, \bar{w}) \leftarrow \mathsf{Extr}(tr, m^*, \sigma^*)$. If $\mathsf{PC}((\bar{p}, m^*), \bar{w}) = 0$ then $\mathcal{B}$ wins. Otherwise, we have $(\bar{p}, m^*) \in \mathcal{L}(\mathsf{PC})$. Thus, by the winning condition of UF, for all i with $Q[i][1] = \bar{p}$ we must have $Q[i][2] \neq \bot$. Thus, in this case too $\mathcal{B}$ wins.

We now deal with adversaries winning UF by outputting $(m^*, \sigma^*) \in Q_S$. $\mathcal{B}$ guesses the index of the first SIGN query involving m^*. (That is, $\mathcal{B}$ picks a random SIGN query and if the queried message has been queried before then $\mathcal{B}$ aborts.) Let the guessed query be for (i^*, m^*, w^*). Instead of querying its own SIGN oracle, $\mathcal{B}$ does the following: If $Q[i^*][2] = \bot$ then $\mathcal{B}$ looks up the signing key sk^* it forwarded when $\mathcal{A}$ queried REVEALSK, otherwise $\mathcal{B}$ queries its own REVEALSK oracle to get sk^*. $\mathcal{B}$ computes $\sigma^* \leftarrow\!\!\$\ \mathsf{Sign}(sk^*, m^*, w^*)$ and returns σ^*. Every time $\mathcal{A}$ makes a SIGN query involving m^*, $\mathcal{B}$ also proceeds as just described. It is clear that this perfectly simulates $\mathcal{A}$'s oracles in UF.

Assume now that $\mathcal{A}$ wins by outputting (m^*, σ^*) and that $\mathcal{B}$ guessed correctly. Since $\mathcal{A}$ wins, we have (1) $\mathsf{Verify}(pp, m^*, \sigma^*) = 1$. Since $\mathcal{B}$ guessed correctly, no entry in Q_S starts with m^*, thus (2) $(m^*, \sigma^*) \notin Q_S$. Let $(\bar{p}, \bar{w}) \leftarrow \mathsf{Extr}(tr, m^*, \sigma^*)$. If no key for $\bar{p}$ was ever revealed then $\mathcal{B}$ wins. There are two ways that a key for $\bar{p}$ could have been revealed in UF-(1): either $\mathcal{A}$ queried REVEALSK for it, or $\mathcal{A}$ made a SIGN query involving m^* (in which case $\mathcal{B}$ may have queried REVEALSK). In either case, since $\mathcal{A}$ won UF, by its winning condition, we must have $(\bar{p}, m^*) \notin \mathcal{L}(\mathsf{PC})$. A forteriori $\mathsf{PC}(\bar{p}, m^*), \bar{w}) = 0$; thus, also in the case that keys for $\bar{p}$ were revealed, $\mathcal{B}$ wins.

Claim. EXT $\wedge$ SIM $\Rightarrow$ UF-(1).

By SIM, UF-(1) is indistinguishable from $\mathbf{Exp}^{\text{UF-(2)}}_{\mathcal{PBS}}$, given in Figure 8, where we replaced Setup by $\mathsf{SimSetup}$, KeyGen by $\mathsf{SKeyGen}$ and Sign by $\mathsf{SimSign}$ conditioned on the fact that the inputs satisfy PC. Note that it is because both games are efficiently decidable, that we can construct a distinguisher which uses an adversary behaving differently in the two games to break SIM.

It is now easy to see that any $\mathcal{A}$ winning UF-(2) can be used to construct $\mathcal{B}$ winning EXT. When $\mathcal{A}$ queries MAKESK(p), $\mathcal{B}$ simply stores $(p, \emptyset, \emptyset)$ in Q. When $\mathcal{A}$ queries REVEALSK(i), $\mathcal{B}$ queries SKEYGEN$(Q[i][1])$. And whenever $\mathcal{A}$ queries SIGN(i, m, w), if $\mathsf{PC}((Q[i][1], m), w) = 1$ then $\mathcal{B}$ queries SIMSIGN(m). $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins.

B Security Proofs for the Instantiation in Section 4.1

SIMULATABILITY. We show that two runs of $\mathbf{Exp}^{\text{SIM}}_{\mathcal{PBS},\mathcal{A}}$, one with b set to 1, and one with b set to 0 are indistinguishable. We proceed by games: We start with $\mathbf{Exp}^{\text{SIM}|b=1}_{\mathcal{PBS},\mathcal{A}}$. Our first change is that we replace $crs \leftarrow\!\!\$\ \mathsf{Setup}_{\text{nizk}}(1^\lambda)$ by $crs \leftarrow\!\!\$\ \mathsf{SimSetup}_{\text{nizk}}(1^\lambda)$ and in calls to SIGNATURE, we replace π by a simulated proof. This is indistinguishable by the zero-knowledge property of $\mathcal{NIZK}$. Note that in this game, the

[5] We note that winning UF this way can only happen if there exist p^*, m^* and w^*, with $\mathsf{PC}((p^*, m^*), w^*) = 0$, but for which $\mathsf{Sign}(pp, \mathsf{KeyGen}(pp, msk, p^*), m^*, w^*)$ outputs a valid signature, which we have not explicitly excluded. However, by this very attack, this cannot happen in any scheme satisfying UF; neither in any scheme satisfying EXT and SIM, since we prove that they imply UF.

proof π can be computed knowing only $(pk, mvk, C_p, C_s, C_w, ovk, m)$, but not the content of C_p, C_s and C_w nor their randomness.

In the next game, for the SIGNATURE calls, we replace the ciphertexts C_p and C_w by encryptions of 0 and C_s by an encryption of a signature $s \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{sig}}(msk, 0\|ovk)$. This is indistinguishable by IND-CPA of $\mathcal{PKE}$. In the final game, we replace the simulated CRS by a real CRS. This final game is $\mathbf{Exp}^{\mathrm{SIM}|b=0}_{\mathcal{PBS},\mathcal{A}}$, which concludes the proof.

EXTRACTABILITY: This notion is reduced to the security of both $\mathcal{S}ig$ and $\mathcal{O}t\mathcal{S}ig$. We distinguish two types of adversaries. Type 1 returns a forgery $(m, (ovk, C_p, C_s, C_w, \pi, \tau))$ such that there was a reply of a SIMSIGN query with the same ovk, Type-2 adversaries return forgeries with a fresh ovk.

We use Type-1 adversaries to break strong unforgeability of $\mathcal{O}t\mathcal{S}ig$. Given ovk^* from our challenger, we simulate $\mathbf{Exp}^{\mathrm{EXT}}_{\mathcal{PBS},\mathcal{A}}$ except that for a randomly guessed valid SIMSIGN query, say for $\hat{m}$, we use ovk^*, compute $\hat{C}_p$, $\hat{C}_s$, $\hat{C}_w$ and $\hat{\pi}$ as per $\mathsf{SimSign}$ and complete the signature by querying our one-time oracle on $(\hat{m}, \hat{C}_p, \hat{C}_s, \hat{C}_w, \hat{\pi})$ to get $\hat{\tau}$. Assume that the adversary $\mathcal{A}$ is of Type 1, then with non-negligible probability his forgery is of the form $(m, \sigma = (ovk^*, C_p, C_s, C_w, \pi, \tau))$. Since, for $\mathcal{A}$ to have won, (m, σ) must be different from all query/response pairs for SIMSIGN, in particular, it must be different from $(\hat{m}, (ovk^*, \hat{C}_p, \hat{C}_s, \hat{C}_w, \hat{\pi}, \hat{\tau}))$, in which we embedded ovk^*. Therefore $((m, C_p, C_s, C_w, \pi), \tau) \neq ((\hat{m}, \hat{C}_p, \hat{C}_s, \hat{C}_w, \hat{\pi}), \hat{\tau})$ and τ is a one-time forgery on (m, C_p, C_s, C_w, π).

We now use Type-2 adversaries to break unforgeability of $\mathcal{S}ig$. On receiving vk^* from our challenger, we run $crs \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{nizk}}(1^\lambda)$ and $(pk, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{pke}}(1^\lambda)$. We run the adversary $\mathcal{A}$ on $pp := (crs, pk, vk^*)$. When $\mathcal{A}$ makes an SKEYGEN query, we compute sk_p by querying our signing oracle on $1\|p$; when $\mathcal{A}$ makes a SIMSIGN query, we use our oracle to get a signature s on $0\|ovk$ and use that as a witness for the proof π in our PBS σ.

Suppose $\mathcal{A}$ wins the game by outputting $(m, \sigma = (ovk, C_p, C_s, \pi, \tau))$, which satisfies Verify. Using Extr, we get $p \leftarrow \mathsf{Dec}(dk, C_p)$ and $w \leftarrow \mathsf{Dec}(dk, C_w)$; additionally, we compute $s \leftarrow \mathsf{Dec}(dk, C_s)$. As σ is valid, soundness of $\mathcal{NIZK}$ and correctness of $\mathcal{PKE}$ imply that either (i) s is a valid signature on $1\|p$ and $\mathsf{PC}((p, m), w) = 1$; or (ii) s is a valid signature on $0\|ovk$. In case (ii), s is a valid forgery on $0\|ovk$, since Type-2 adversaries use a one-time key which was never issued in a SIMSIGN query. In case (i), since $\mathcal{A}$ wins and $\mathsf{PC}((p, m), w) = 1$, we must have $p \notin Q_K$, thus s is a valid forgery on $1\|p$.

C Proofs for the Construction of Attribute-Based Signatures

PRIVACY. Let $\mathcal{A}$ be an adversary against privacy of $\mathcal{ABS}$. It is quite straightforward to build $\mathcal{B}$ breaking indistinguishability of $\mathcal{PBS}$. $\mathcal{B}$ receives (pp, msk) from its challenger and forwards them to $\mathcal{A}$ as (pp, ask). When $\mathcal{A}$ queries $\mathrm{LR}(A_0, A_1, m, \Upsilon)$, $\mathcal{B}$ queries its own LR oracle for the two policies $p_0 := A_0$ and $p_1 := A_1$, and the message (Υ, m), and forwards the response (σ, sk_0, sk_1) to $\mathcal{A}$.

For $i \in \{0, 1\}$, we have $\mathsf{PC}(p_i, (\Upsilon, m)) = 0$ iff $\Upsilon(A_i) = 0$, thus $\mathcal{B}$'s oracle returns $\perp$, whenever $\mathcal{A}$'s oracle should return $\perp$. Otherwise $\mathcal{B}$'s oracle computes $sk_i \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{pbs}}(msk, p_i)$, for $i \in \{0, 1\}$ and $\sigma_b \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{pbs}}(sk_b, (\Upsilon, m))$, which, by definition is the same as $sk_i \leftarrow\!\!\$\ \mathsf{AttrGen}(ask, A_i)$, for $i \in \{0, 1\}$, and $\sigma_b \leftarrow\!\!\$\ \mathsf{Sign}(sk_b, m, \Upsilon)$. Thus $\mathcal{A}$ receives what it expects from its oracle LR. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. It is clear that $\mathcal{B}$ guesses b correctly whenever $\mathcal{A}$ does; thus, $\mathbf{Adv}^{\mathrm{IND}}_{\mathcal{PBS},\mathcal{B}} = \mathbf{Adv}^{\mathrm{PRIV}}_{\mathcal{ABS},\mathcal{A}}$.

UNFORGEABILITY. Assume $\mathcal{A}$ wins $\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{ABS}}$; we construct $\mathcal{B}$ which wins $\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{PBS}}$ with the same probability. $\mathcal{B}$ receives pp, forwards it to $\mathcal{A}$ and answers $\mathcal{A}$'s oracle queries as follows, after initializing a counter $j \leftarrow 1$. ATTRGEN(A): $\mathcal{B}$ queries MAKESK(A) and REVEALSK(j), forwards the received key sk_A to $\mathcal{A}$ and sets $j \leftarrow j + 1$. SIGN(m, Υ): $\mathcal{B}$ picks a random $A \subseteq \mathbb{A}$ with $\Upsilon(A) = 1$, queries MAKESK(A) and SIGN($j, (\Upsilon, m)$), forwards the received signature to $\mathcal{A}$ and sets $j \leftarrow j + 1$. When $\mathcal{A}$ outputs (m, Υ, σ), $\mathcal{B}$ outputs $((\Upsilon, m), \sigma)$. Suppose $\mathcal{A}$ wins by outputting (m, Υ, σ). Then $1 = \mathsf{Verify}(pp, m, \Upsilon, \sigma) = \mathsf{Verify}_{\mathrm{pbs}}(pp, (\Upsilon, m), \sigma)$. Since $\mathcal{A}$ won, we have (1) $(m, \Upsilon) \notin Q_S$ and (2) for all $A \in Q_K : \Upsilon(A) = 0$. Thus in $\mathbf{Exp}^{\mathrm{UF}}_{\mathcal{PBS}}$, we have that for all i, for which $\mathsf{PC}(Q[i][1], (\Upsilon, m)) = \Upsilon(A) = 1$:

proc INITIALIZE // $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pke}}(1^\lambda)$; $Q_C \leftarrow \emptyset$; $Q_S \leftarrow \emptyset$
For $i = 1, \ldots, n$ do
  $sk_i \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pbs}}(msk, (ek, i))$
  $\boldsymbol{gsk}[i] \leftarrow (pp, ek, i, sk_i)$
Return $((pp, ek), dk)$

proc FINALIZE$(m, (c, \sigma))$ // $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$
If $\mathsf{Verify}(pp, (c, m), \sigma) = 0$ then return false
$i^* \leftarrow \mathsf{Dec}(dk, c)$
If $i^* \in [n]$, $i^* \notin Q_C$ and $(i^*, m) \notin Q_S$ then return true
Return false

proc CORRUPT(i) // $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$
$Q_C \leftarrow Q_C \cup \{i\}$
Return $\boldsymbol{gsk}[i]$

proc GSIG(i, m) // $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$
$Q_S \leftarrow Q_S \cup \{(i, m)\}$
$r \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c \leftarrow \mathsf{Enc}(ek, i; r)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(sk_i, (c, m), r)$
Return (c, σ)

Figure 9: Games $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$ for the proof of traceability of $\mathcal{GS}$

$Q[i][2] \neq \perp$ (as otherwise $Q[i][1]$ would be in Q_K), and $(\Upsilon, m) \notin Q[i][3]$ (as otherwise (m, Υ) would be in Q_S). Thus, whenever $\mathcal{A}$ wins $\mathbf{Exp}^{\text{UF}}_{\mathcal{ABS}}$, $\mathcal{B}$ wins $\mathbf{Exp}^{\text{UF}}_{\mathcal{PBS}}$.

D Proofs for the Construction of Group Signatures

TRACEABILITY FROM UF. Plugging in the definition of $\mathcal{GS}[\mathcal{PKE}, \mathcal{PBS}]$ in $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS},\mathcal{A}}$, we get (after some simplification) the experiment in Figure 9.

We build an adversary $\mathcal{B}$ against UF of $\mathcal{PBS}$ which simulates the above game and wins $\mathbf{Exp}^{\text{UF}}_{\mathcal{PBS}}$ whenever $\mathcal{A}$ wins $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$.

$\mathcal{B}(pp : \text{MAKESK}(\cdot), \text{REVEALSK}(\cdot), \text{SIGN}(\cdot,\cdot,\cdot))$
  $(ek, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\text{pke}}(1^\lambda)$
  For $i = 1, \ldots, n$ do
    MAKESK$((ek, i))$
  $(m, (c, \sigma)) \leftarrow\!\!\$\ \mathcal{A}((pp, ek), dk) : \text{CORRUPT}_{\mathcal{B}}(\cdot), \text{GSIG}_{\mathcal{B}}(\cdot,\cdot))$
  Return $(m, (c, \sigma))$

CORRUPT$_{\mathcal{B}}(i)$
  $sk_i \leftarrow$ REVEALSK(i); return (pp, ek, i, sk_i)

GSIG$_{\mathcal{B}}(i, m)$
  $r \leftarrow\!\!\$\ \{0,1\}^\lambda$; $c \leftarrow \mathsf{Enc}(ek, i; r)$
  $\sigma \leftarrow\!\!\$$ SIGN$(i, (c, m), r)$; return (c, σ)

Suppose that $\mathcal{A}$ wins $\mathbf{Exp}^{\text{TRC}}_{\mathcal{GS}}$ by outputting $(m^*, (c^*, \sigma^*))$. Thus $\mathsf{Verify}(pp, (c^*, m^*), \sigma^*) = 1$ and with $i^* \leftarrow \mathsf{Dec}(sk, c^*)$, we have that $i^* \in [n]$ and $\mathcal{A}$ has queried neither CORRUPT(i^*) nor GSIG(i^*, m^*). This means that $\mathcal{B}$ has neither queried REVEALSK(i^*) nor SIGN$(i^*, (c, m^*), r)$ for any c and r, thus $(*)$ $Q[i^*][2] \neq \perp$ and for all c: $(c, m^*) \notin Q[i][3]$, thus in particular $(c^*, m^*) \notin Q[i][3]$.

$\mathcal{B}$ wins $\mathbf{Exp}^{\text{UF}}_{\mathcal{PBS}}$ if $\mathsf{Verify}(pp, (c^*, m^*), \sigma^*) = 1$ and if for all i, for which $(Q[i][1], (c^*, m^*)) \in \mathcal{L}(\mathsf{PC})$, we have (1) $Q[i][2] \neq \perp$ and (2) $(c^*, m^*) \notin Q[i][3]$. Now for any (c, m), we have: $(Q[i][1], (c, m)) \in \mathcal{L}(\mathsf{PC})$ iff c is in the range of $\mathsf{Enc}(ek, i; \cdot)$, which, by correctness of $\mathcal{PKE}$, implies $i = \mathsf{Dec}(dk, c)$. Thus we only have to show (1) and (2) for $i \leftarrow i^*$; which we have already done in $(*)$.

ANONYMITY. We let $\mathbf{Exp}^{\text{anon-}d}_{\mathcal{GS}}$ denote the experiment $\mathbf{Exp}^{\text{ANON}}_{\mathcal{GS}}$ when the bit b is fixed to $b \leftarrow d$. To show anonymity it then suffices to prove that for all PT adversaries $\mathcal{A}$ the difference

$$\left| \Pr[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \Pr[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] \right|$$

is negligible in λ, which we will show by a series of game hops. Plugging our scheme into the definition $\mathbf{Exp}^{\text{anon-}b}_{\mathcal{GS}}$ (see Figure 7), we get the game given on the left of Figure 10, when we ignore the boxes.

We modify this game to $\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS}}$ by including the lines in boxes, which replace each respective preceding line. We show that $\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS}}$ is indistinguishable from $\mathbf{Exp}^{\text{anon-}b}_{\mathcal{GS}}$: An adversary which

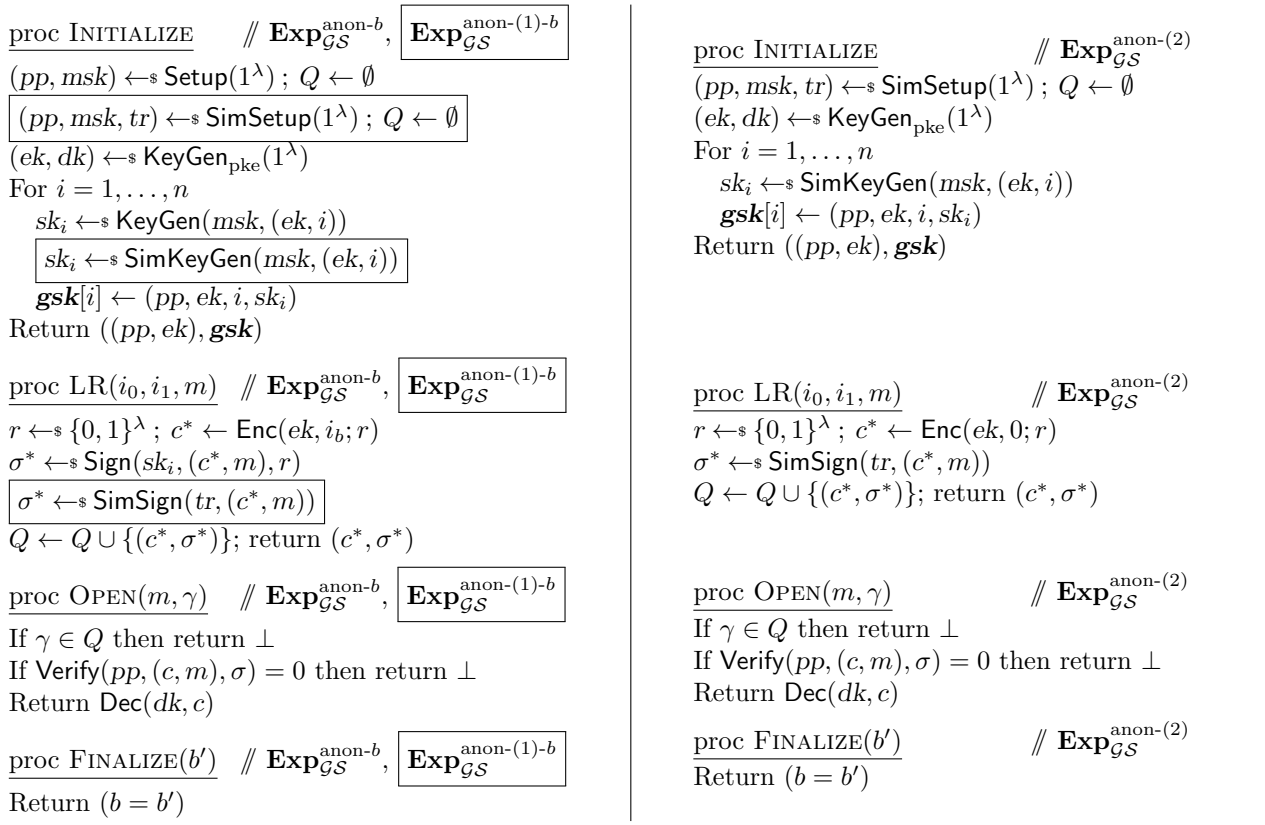

Figure 10: Games $\mathbf{Exp}^{\text{anon-}b}_{\mathcal{GS}}$, $\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS}}$ and $\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS}}$ for the proof of anonymity of $\mathcal{GS}$

behaved differently in the two games could be used to break the notion SIM of $\mathcal{PBS}$ by building an adversary $\mathcal{B}$ as follows: $\mathcal{B}$ gets (pp, msk) from its challenger and uses its oracle Key to compute sk_i and oracle Signature to compute σ in the boxed lines (or the ones above, depending on which game $\mathcal{B}$ plays). (Note that when $\mathcal{B}$ makes a Signature$(i, (c, m), r)$ query when answering the LR query then by definition we have $\mathsf{PC}\big(((ek, i), (c, m)), r\big) = 1$, so Signature returns the output of either $\mathsf{SimSign}$ or Sign.)

We next define $\mathbf{Exp}^{\text{anon-(2)}}$, given on the right in Figure 10, in which c is an encryption of 0 instead of i_b. Note that this final game is independent of the bit b, thus the adversary's probability of outputting b is $\frac{1}{2}$. We show that by IND-CCA of $\mathcal{PKE}$ and EXT of $\mathcal{PBS}$, $\mathbf{Exp}^{\text{anon-(1)-}b}$ and $\mathbf{Exp}^{\text{anon-(2)}}$ are indistinguishable. In Figure 11 we define an adversary $\mathcal{B}^{(b)}$ against $\mathcal{PKE}$ which uses an adversary $\mathcal{A}$ that behaves differently in these two games to break IND-CCA of $\mathcal{PKE}$.

We first define the event QRY marked in the description of $\mathcal{B}^{(b)}$: QRY_b denotes the event that in $\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS}[\mathcal{PKE},\mathcal{PBS}],\mathcal{A}}$, $\mathcal{A}$ makes a valid Open query $(m, (c^*, \sigma))$ with $\sigma \neq \sigma^*$. QRY_z denotes the event that $\mathcal{A}$ does so in $\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS}[\mathcal{PKE},\mathcal{PBS}],\mathcal{A}}$, i.e., when c^* is an encryption of 0. In our analysis, we require the following lemma, which we prove below.

Lemma 1. *The probability that the event* QRY_z *happens is upper-bounded by* $\mathbf{Adv}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]}$ *with* $\mathcal{B}_E[\mathcal{A}]$ *defined in Figure 11.*

We now show that if $\mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{PKE},\mathcal{B}^{(b)}[\mathcal{A}]}$ and $\mathbf{Exp}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]}$ are negligible then so is $\mathbf{Adv}^{\text{anon-(1)}}_{\mathcal{GS}[\mathcal{PKE},\mathcal{PBS}],\mathcal{A}}$, which implies anonymity of our group-signature scheme. Analyzing the behavior of $\mathcal{B}^{(b)}$ in $\mathbf{Exp}^{\text{IND-CCA-}d}_{\mathcal{PKE}}$,

$\underline{\mathcal{B}^{(b)}(ek : \mathrm{LR}(\cdot,\cdot), \textsc{Dec}(\cdot))}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$
For $i = 1, \ldots, n$
$sk_i \leftarrow\!\!\$\ \mathsf{SimKeyGen}(msk, (ek, i))$
$\boldsymbol{gsk}[i] \leftarrow (pp, ek, i, sk_i)$
$b' \leftarrow\!\!\$\ \mathcal{A}((pp, ek), \boldsymbol{gsk} : \mathrm{LR}_{\mathcal{B}^{(b)}}(\cdot,\cdot,\cdot), \textsc{Open}_{\mathcal{B}^{(b)}}(\cdot,\cdot))$
Return b'

$\underline{\mathrm{LR}_{\mathcal{B}^{(b)}}(i_0, i_1, m)}$
$c^* \leftarrow\!\!\$\ \mathrm{LR}(0, i_b)$
$\sigma^* \leftarrow\!\!\$\ \mathsf{SimSign}(tr, (c^*, m))$
Return (c^*, σ^*)

$\underline{\textsc{Open}_{\mathcal{B}^{(b)}}(m, (c, \sigma))}$
If $\mathsf{Verify}(pp, (c, m), \sigma) = 0$ then return $\perp$
If $c = c^*$ and $\sigma \neq \sigma^*$ (QRY)
$\mathcal{B}$ halts and returns 1
Return $\textsc{Dec}(c)$

$\underline{\mathcal{B}_E(pp : \textsc{SKeyGen}(\cdot), \textsc{SimSign}(\cdot,\cdot,\cdot))}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{pke}}(1^\lambda)$
For $i = 1, \ldots, n$
$sk_i \leftarrow\!\!\$\ \textsc{SKeyGen}((ek, i))$
$\boldsymbol{gsk}[i] \leftarrow (pp, ek, i, sk_i)$
$b' \leftarrow\!\!\$\ \mathcal{A}((pp, ek), \boldsymbol{gsk} : \mathrm{LR}_{\mathcal{B}_E}(\cdot,\cdot,\cdot), \textsc{Open}_{\mathcal{B}_E}(\cdot,\cdot))$
Return b'

$\underline{\mathrm{LR}_{\mathcal{B}_E}(i_0, i_1, m)}$
$r^* \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c^* \leftarrow \mathsf{Enc}(ek, 0; r^*)$
$\sigma^* \leftarrow\!\!\$\ \textsc{SimSign}((c^*, m))$
Return (c^*, σ^*)

$\underline{\textsc{Open}_{\mathcal{B}_E}(m, (c, \sigma))}$
If $\mathsf{Verify}(pp, (c, m), \sigma) = 0$ then return $\perp$
If $c = c^*$ and $\sigma \neq \sigma^*$ (QRY)
$\mathcal{B}_E$ halts and returns $((c^*, m), \sigma)$
Return $\textsc{Dec}(c)$

Figure 11: Adversaries $\mathcal{B}^{(b)}$ against IND-CCA of $\mathcal{PKE}$ and $\mathcal{B}_E$ against EXT of $\mathcal{PBS}$

for $d = 0, 1$, we get:

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{IND-CCA-1}}_{\mathcal{PKE},\mathcal{B}^{(b)}[\mathcal{A}]} = 1] &= \Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1 \wedge \neg\mathsf{QRY}_b] + \Pr[\mathsf{QRY}_b] \\ &\geq \Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1]\end{aligned}$$

and

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{IND-CCA-0}}_{\mathcal{PKE},\mathcal{B}^{(b)}[\mathcal{A}]} = 1] &= \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1 \wedge \neg\mathsf{QRY}_z] + \Pr[\mathsf{QRY}_z] \\ &\leq \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1] + \mathbf{Adv}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]}\end{aligned}$$

Together this yields

$$\Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1] \;\leq\; \mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{PKE},\mathcal{B}^{(b)}[\mathcal{A}]} + \mathbf{Adv}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]} \tag{8}$$

It remains now to lower-bound the probability on the left-hand side of (8). Let $\overline{\mathcal{B}}^{(b)}$ be defined as $\mathcal{B}^{(b)}$ but behaving like $\mathcal{B}^{(b)}$ does in $\mathbf{Exp}^{\text{IND-CPA-}(1-b)}$, that is: when answering $\mathcal{A}$'s $\mathrm{LR}_{\mathcal{B}}$ query, it queries its own LR oracle on $(i_b, 0)$; moreover, if during an $\textsc{Open}$ query the event QRY happens, it returns 0 (rather than 1). We then get:

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{IND-CCA-1}}_{\mathcal{PKE},\overline{\mathcal{B}}^{(b)}[\mathcal{A}]} = 1] &= \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1 \wedge \neg\mathsf{QRY}_z] \\ &= \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1 \wedge \mathsf{QRY}_z] \\ &\geq \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1] - \Pr[\mathsf{QRY}_z]\end{aligned}$$

and

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{IND-CCA-0}}_{\mathcal{PKE},\overline{\mathcal{B}}^{(b)}[\mathcal{A}]} = 1] &= \Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1 \wedge \neg\mathsf{QRY}_b] \\ &\leq \Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1]\end{aligned}$$

Together this yields

$$\Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1] \;\leq\; \mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{PKE},\overline{\mathcal{B}}^{(b)}[\mathcal{A}]} + \mathbf{Adv}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]} \;. \tag{9}$$

(8) and (9) together yield:

$$\begin{aligned}\big|\Pr[\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS},\mathcal{A}} = 1]\big| \\ \leq\; \max\big\{\mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{PKE},\mathcal{B}^{(b)}[\mathcal{A}]}, \mathbf{Adv}^{\text{IND-CCA}}_{\mathcal{PKE},\overline{\mathcal{B}}^{(b)}[\mathcal{A}]}\big\} + \mathbf{Adv}^{\text{EXT}}_{\mathcal{PBS},\mathcal{B}_E[\mathcal{A}]} \;,\end{aligned}$$

thus, assuming $\mathcal{PKE}$ satisfies IND-CCA and $\mathcal{PBS}$ satisfies EXT, the games $\mathbf{Exp}^{\text{anon-(1)-}b}_{\mathcal{GS}}$ and $\mathbf{Exp}^{\text{anon-(2)}}_{\mathcal{GS}}$ are indistinguishable. Since the last game is independent of the bit b, together we have that games $\mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS}[\mathcal{PKE},\mathcal{PBS}],\mathcal{A}}$ and $\mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS}[\mathcal{PKE},\mathcal{PBS}],\mathcal{A}}$ are indistinguishable. It remains to prove Lemma 1.

Proof of Lemma 1. We construct an adversary $\mathcal{B}_E$ which breaks EXT whenever QRY happens in the game where $\mathcal{A}$ is given as c^* an encryption of 0. $\mathcal{B}_E$ is given on the right of Figure 11 and perfectly simulates $\mathbf{Exp}^{\text{IND-CCA-0}}_{\mathcal{PKE},\mathcal{B}[\mathcal{A}]}$.

We show that whenever the event QRY happens then $\mathcal{B}_E$ wins $\mathbf{Exp}^{\text{EXT}}_{\mathcal{PBS}}$ by returning $((c^*, \hat{m}), \hat{\sigma})$: We have to show that in this case we have (1) $\mathsf{Verify}(pp, (c^*, \hat{m}), \hat{\sigma}) = 1$; (2) $((c^*, \hat{m}), \hat{\sigma})$ is not among the query/response pairs for SimSign calls; and (3) with $((\hat{ek}, \hat{\imath}), \hat{r}) \leftarrow \mathsf{Extr}(tr, (c^*, \hat{m}), \hat{\sigma})$ we have either (3a) $(\hat{ek}, \hat{\imath})$ was never queried to SKeyGen or (3b) $\mathsf{PC}\big(((\hat{ek}, \hat{\imath}), (c^*, \hat{m})), \hat{r}\big) = 0$.

(1) is the case as otherwise $\mathcal{B}_E$ would not have halted in the $\textsc{Open}_{\mathcal{B}}$ query. (2) is satisfied, as σ^* is the only reply of a SimSign call, and by (QRY) we have $\hat{\sigma} \neq \sigma^*$. Suppose (3a) is not satisfied, which means $\hat{ek}$ is the encryption key set up by $\mathcal{B}_E$ and $1 \leq \hat{\imath} \leq n$. By Equation (7), Condition (3b) means $c^* \neq \mathsf{Enc}(ek, \hat{\imath}; \hat{r})$. This is satisfied, since $c^* = \mathsf{Enc}(ek, 0; r^*)$ and by correctness of $\mathcal{PKE}$, c^* cannot be the encryption of a different message. □

E Other Primitives Implied by PBS

Theorem 2 shows which primitives are sufficient for policy-based signatures. We now show which primitives are necessary, that is, which fundamental cryptographic primitives are implied by PBS. For ease of readability, we restate the construction from Section 7.

Simulation-sound extractable NIZK proofs. Groth [Gro06] introduced a notion of simulation-soundness for NIZK proofs of knowledge. It requires that even when an adversary is provided with an oracle producing simulated proofs for (not necessarily true) statements of his choice, the adversary cannot produce a new valid proof from which the extractor fails to extract a witness. The formal definitions can be found in the full version of [Gro06]. We simplify them slightly, in that we do not distinguish between the trapdoors for simulation and extraction, and thus only have one $\mathsf{SimSetup}$ algorithm:

Let $\mathbf{Exp}^{\text{ZK}}_{\mathcal{NIZK}}$ and $\mathbf{Exp}^{\text{SSE}}_{\mathcal{NIZK}}$ be defined in Figure 12. A proof system $\mathcal{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for a relation R is *zero-knowledge* if there exist $\mathsf{SimSetup}$ and $\mathsf{SimSign}$ such that $\mathbf{Adv}^{\text{ZK}}_{\mathcal{NIZK},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\text{ZK}}_{\mathcal{NIZK},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}] - \frac{1}{2}$ is negligible in λ for all PT $\mathcal{A}$. It is *simulation-sound extractable* if there exists Extr such that $\mathbf{Adv}^{\text{SSE}}_{\mathcal{NIZK},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\text{SSE}}_{\mathcal{NIZK},\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}]$ is negligible in λ for all PT $\mathcal{A}$.

Let R be an **NP**-relation and let PC and p^* be such that $\mathsf{PC}((p^*, x), w) = \mathsf{R}(x, w)$. Let $\mathcal{PBS} = (\mathsf{Setup}_{\text{pbs}}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}_{\text{pbs}}, \mathsf{SimSetup}_{\text{pbs}}, \mathsf{SKeyGen}, \mathsf{SimSign}, \mathsf{Extr}_{\text{pbs}})$ be a policy-based signature scheme for PC which satisfies simulatability and extractability. Then the following scheme $\mathcal{NIZK}[\mathcal{PBS}]$ is a simulation-sound extractable NIZK proof system:

$\mathbf{Exp}^{\mathrm{ZK}}_{\mathcal{NIZK}}$

proc INITIALIZE
$b \leftarrow \{0,1\}$
$(crs_0, tr) \leftarrow \mathsf{SimSetup}_{\mathrm{zk}}(1^\lambda)$
$(crs_1) \leftarrow \mathsf{Setup}_{\mathrm{zk}}(1^\lambda)$
Return crs_b

proc PROOF(x, w)
If $\mathsf{R}(x, w) = 1$ then $\pi_0 \leftarrow\!\!\$\ \mathsf{SimProve}(tr, x)$
else $\pi_0 \leftarrow \bot$
$\pi_1 \leftarrow\!\!\$\ \mathsf{Prove}(crs_1, x, w)$
Return π_b

proc FINALIZE(b')
Return $(b = b')$

$\mathbf{Exp}^{\mathrm{SSE}}_{\mathcal{NIZK}}$

proc INITIALIZE
$(crs, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}(1^\lambda)$; $Q \leftarrow \emptyset$
return crs

proc SIMPROVE(x)
$\pi \leftarrow\!\!\$\ \mathsf{SimProve}(crs, tr, x)$; $Q \leftarrow Q \cup \{(x, \pi)\}$
return π

proc FINALIZE(x, π)
$w \leftarrow \mathsf{Extr}(tr, x, \pi)$
Return 1 if all of the following hold:
$(x, \pi) \notin Q$
$\mathsf{Verify}(crs, x, \pi) = 1$
$\mathsf{R}(x, w) = 0$

Figure 12: Games defining zero-knowledge and simulation-sound extractability for NIZK

$\underline{\mathsf{Setup}_{\mathrm{zk}}(1^\lambda)}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{pbs}}(1^\lambda)$
$sk \leftarrow \mathsf{KeyGen}(msk, p^*)$
Return $crs \leftarrow (pp, sk)$

$\underline{\mathsf{Prove}((pp, sk), x, w)}$
Return $\pi \leftarrow \mathsf{Sign}(sk, x, w)$

$\underline{\mathsf{Verify}_{\mathrm{zk}}((pp, sk), x, \pi)}$
Return $\mathsf{Verify}_{\mathrm{pbs}}(pp, x, \pi)$

$\underline{\mathsf{Extr}_{\mathrm{zk}}(tr, x, \pi)}$
$(p, w) \leftarrow \mathsf{Extr}_{\mathrm{pbs}}(tr, x, \pi)$
Return w

$\underline{\mathsf{SimSetup}_{\mathrm{zk}}(1^\lambda)}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}_{\mathrm{pbs}}(1^\lambda)$
$sk \leftarrow \mathsf{SKeyGen}(tr, p^*)$
Return $crs \leftarrow (pp, sk)$ and tr

$\underline{\mathsf{SimProve}((pp, sk), tr, x)}$
Return $\pi \leftarrow \mathsf{SimSign}(tr, x)$

The proofs that SIM of $\mathcal{PBS}$ implies zero knowledge of $\mathcal{NIZK}$ and that EXT of $\mathcal{PBS}$ implies simulation-sound extractability of $\mathcal{NIZK}$ are straightforward.

PUBLIC-KEY ENCRYPTION. In order to construct a public-key encryption (PKE) scheme from PBS, we note that policies can be interpreted as plaintexts hidden in a signature, which can be "decrypted" using extractability. Consider the following scheme $\mathcal{PKE}[\mathcal{PBS}]$ constructed from $\mathcal{PBS}$ for PC s.t. $\mathsf{PC}((p, m^*), w^*) = 1$ for some m^*, w^* and all p.

$\underline{\mathsf{KeyGen}_{\mathrm{pke}}(1^\lambda)}$
$(pp, msk, tr) \leftarrow\!\!\$\ \mathsf{SimSetup}_{\mathrm{pbs}}(1^\lambda)$
Return $(pk \leftarrow (pp, msk),\ dk \leftarrow tr)$

$\underline{\mathsf{Enc}((pp, msk), x)}$
$sk \leftarrow\!\!\$\ \mathsf{KeyGen}(msk, x)$
Return $c \leftarrow\!\!\$\ \mathsf{Sign}(sk, m^*, w^*)$

$\underline{\mathsf{Dec}(dk, c)}$
$(x, w) \leftarrow \mathsf{Extr}(dk, m^*, c)$
Return x

If $\mathcal{PBS}$ satisfies IND then it follows immediately that $\mathcal{PKE}$ satisfies indistinguishability of ciphertexts under chosen-plaintext attack (IND-CPA).

Sahai [Sah99] shows that when instantiating the Naor-Yung [NY90] construction with a simulation-sound NIZK, one obtains a chosen-ciphertext-attack (CCA)-secure public-key encryption scheme, which is what we require for our construction of group signatures in Section 6. Combining our constructions above of SSE NIZK and CPA PKE, we thus obtain a CCA-secure PKE; which we could also directly construct as follows:

Let $\mathcal{PKE}_{\mathrm{cpa}} = (\mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Enc}_{\mathrm{cpa}}, \mathsf{Dec}_{\mathrm{cpa}})$ be a CPA-secure PKE; let $\mathcal{PBS}$ be a policy-based signature scheme for the policy checker

$$\mathsf{PC}(((pk_0, pk_1), (c_0, c_1)), (x, r_0, r_1)) = 1 \iff c_0 = \mathsf{Enc}_{\mathrm{pke}}(pk_0, x; r_0) \ \wedge\ c_1 = \mathsf{Enc}_{\mathrm{pke}}(pk_1, x; r_1) \qquad (10)$$

Then the following scheme $\mathcal{PKE}_{\mathrm{cca}}$ is an IND-CCA secure PKE scheme:

$\underline{\mathsf{KeyGen}_{\mathrm{cca}}(1^\lambda)}$
$(pk_0, dk_0) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{cpa}}(1^\lambda)$
$(pk_1, dk_1) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{cpa}}(1^\lambda)$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{pbs}}(1^\lambda)$
$sk \leftarrow\!\!\$\ \mathsf{KeyGen}(msk, (pk_0, pk_1))$
Return $\overline{pk} \leftarrow (pk_0, pk_1, sk)$
$\overline{dk} \leftarrow (pp, dk_0)$

$\underline{\mathsf{Enc}_{\mathrm{cca}}((pk_0, pk_1, sk), x)}$
$r_0, r_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$c_0 \leftarrow \mathsf{Enc}_{\mathrm{cpa}}(pk_0, x; r_0)$
$c_1 \leftarrow \mathsf{Enc}_{\mathrm{cpa}}(pk_1, x; r_1)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(sk, (c_0, c_1), (x, r_0, r_1))$
Return $\overline{c} = (c_0, c_1, \sigma)$

$\underline{\mathsf{Dec}_{\mathrm{cca}}((pp, dk_0), (c_0, c_1, \sigma))}$
If $\mathsf{Verify}(pp, (c_0, c_1), \sigma) = 0$
Return $\perp$
Return $\mathsf{Dec}_{\mathrm{cpa}}(dk_0, c_0)$

Signatures of knowledge. Lastly, we show that PBS also imply another related primitive: In signatures of knowledge (SoK) [CL06] there is a setup outputting trusted parameters, but no authority. Anyone can make a signature on a message m w.r.t. an **NP**-relation R and a statement x, if they know a witness w for x, i.e., $R(x, w) = 1$. Security is defined by simulatability and extractability notions analogous to ours for PBS. We refer to [CL06] for the definitions.

To construct a SoK, we use a PBS which signs messages of the form (R, x, m) and require a signer to know a witness w for x w.r.t. R. In particular, let $\mathcal{PBS}$ be a PBS scheme for the policy checker PC s.t. for some policy p^* we have: $\mathsf{PC}((p^*, (R, x, m), w) = 1$ iff $R(x, w) = 1$. Then we define:

$\underline{\mathsf{Setup}_{\mathrm{sok}}(1^\lambda)}$
$(pp, msk) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathrm{pbs}}(1^\lambda)$
$sk \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathrm{pbs}}(msk, p^*)$
Return $pp_{\mathrm{sok}} = (pp, sk)$

$\underline{\mathsf{Sign}_{\mathrm{sok}}((pp, sk), R, x, m, w)}$
$c \leftarrow\!\!\$\ \mathsf{Sign}_{\mathrm{pbs}}(sk, (R, x, m), w)$
Return c

$\underline{\mathsf{Verify}_{\mathrm{sok}}((pp, sk), R, x, m, \sigma)}$
Return $\mathsf{Verify}_{\mathrm{pbs}}(pp, (R, x, m), \sigma)$

$\mathsf{SimSetup}_{\mathrm{sok}}$, $\mathsf{SimSign}_{\mathrm{sok}}$ and $\mathsf{Extr}_{\mathrm{sok}}$, as required by the security definitions are defined by replacing the respective PBS algorithms by their simulated variants. $\mathsf{Extr}_{\mathrm{sok}}$ runs $\mathsf{Extr}_{\mathrm{pbs}}$ to get (p, w) and returns w. It is then straightforward to show that the above scheme satisfies both simulatability and extractability.

F Efficient Instantiations

Group signatures. The efficient instantiation of policy-based signatures given in Section 4.2 requires that the policy checker PC be expressible as a set of pairing-product equations. For our construction of group signatures from PBS in Section 6, this means that Equation (7) must be expressed as an equation of the form given in Equation (2).

We thus need to express the predicate "*is an encryption of my identity*" as a statement in the language of pairing-product equations. The witness therefore must be a group element, so it seems that we need an encryption scheme whose randomness is a group element. It is however sufficient to find a group element which witnesses the fact that a ciphertext is the encryption of a certain plaintext.

Let user identities be elements $I \in \mathbb{G}$ and define the opener's public key as $Y \in \mathbb{G}$. An ElGamal encryption of I under public key Y is defined by choosing $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and setting $(C, D) \leftarrow (I \cdot Y^r, G^r)$. Using the bilinear map, $W \leftarrow H^r$ is a witness for encryption of I using the two equations:

$$e(C, H) = e(I, H)\, e(Y, \underline{W}) \qquad\qquad e(D, H) = e(G, \underline{W}) \tag{11}$$

Thus, for a policy I (which we identify with the user identity) and a message of the form (C, D, M), the policy checker for our group-signature construction is defined as

$$\mathsf{PC} := \{((I, (C, D, M)), W) \mid (I, C, D, W) \text{ satisfy Equation (11)}\} \ .$$

In order to efficiently instantiate our group-signature construction given in Section 6, we thus require a CCA-secure encryption, which contains as part of a ciphertext an ElGamal ciphertext, which we show how to construct next.

CCA-SECURE ENCRYPTION. In Section 7, we showed how to combine two ciphertexts and a policy-based signature on them to a CCA-secure public-key encryption. The policy checker for the PBS is defined in Equation (10). Using as the CPA-secure scheme ElGamal encryption, this can be expressed as a set of pairing-product equations, where $Y_1, Y_2 \in \mathbb{G}$ are be the public keys defining the policy, the ciphertexts $(C_0, D_0) \in \mathbb{G}^2$ and $(C_1, D_1) \in \mathbb{G}^2$ represent the PBS-message and $(X, W_0, W_1) \in \mathbb{G} \times \mathbb{H}^2$ is the witness:

$$e(C_0, H) = e(\underline{X}, H)\, e(Y_0, \underline{W}_0) \qquad e(D_0, H) = e(G, \underline{W}_0)$$
$$e(C_1, H) = e(\underline{X}, H)\, e(Y_1, \underline{W}_1) \qquad e(D_1, H) = e(G, \underline{W}_1)$$