Accountable Metadata-Hiding Escrow: A Group Signature Case Study

Citation for published version:
Kohlweiss, M & Miers, I 2015, 'Accountable Metadata-Hiding Escrow: A Group Signature Case Study', *Water Treatment Technology*, vol. 2015, no. 2, pp. 206-221. https://doi.org/10.1515/popets-2015-0012.

Digital Object Identifier (DOI):
10.1515/popets-2015-0012.

Published In:
Water Treatment Technology

DE GRUYTER OPEN Proceedings on Privacy Enhancing Technologies 2015; 2015 (2):206–221

Markulf Kohlweiss and Ian Miers*

Accountable Metadata-Hiding Escrow: A Group Signature Case Study

Abstract: A common approach to demands for lawful access to encrypted data is to allow a trusted third party (TTP) to gain access to private data. However, there is no way to verify that this trust is well placed as the TTP may open all messages indiscriminately. Moreover, existing approaches do not scale well when, in addition to the content of the conversation, one wishes to hide one's identity. Given the importance of metadata this is a major problem. We propose a new approach in which users can retroactively verify cryptographically whether they were wiretapped. As a case study, we propose a new signature scheme that can act as an accountable replacement for group signatures, *accountable forward and backward tracing signatures*.

Keywords: Accountability, traceable signatures, group signatures

DOI 10.1515/popets-2015-0012
Received 2015-02-15; revised 2015-05-18; accepted 2015-05-18.

1 Introduction

For digital communication there are few scaling issues to prevent mass surveillance and, indeed, no easy way to detect it. To ensure privacy, we rely on cryptographic guarantees. These are one of the best tools to prevent such surveillance: even if there is a "political" solution to one's own government's spying, there are always other governments, and in both cases there is the added difficulty of actually verifying that (covert) surveillance isn't occurring.

Cryptographic protections, on the other hand, are absolute. However, precisely because they are inviolable, widespread deployment of such systems, e.g., in cloud services, often raises governmental objections or requires mandated ways of providing access. A commonly proposed solution for lawful access to encrypted communications is to appoint a trusted party (or multiple partially-trusted parties) who escrow(s) a user's identity or information about her actions for future retrieval. This has been applied to anonymous signatures [8, 10], e-cash schemes [9], and even saw very limited (and ultimately failed) real-world deployment in the form of the Clipper chip for encryption. Indeed, the US and several other countries are currently trying to mandate the adoption of such techniques. However, especially in light of recent revelations about the intelligence industry, it should be clear that such proposals make cryptographic protections against mass surveillance worthless:

First, in the face of nation-states that are willing to compromise hardware, penetrate systems, and coerce individuals, such an approach seems foolhardy. Eventually the trusted party's key or unfettered access to it will be extracted even if the party is itself trustworthy.

Second, escrow systems typically fail to hide metadata: users' messages are encrypted with an escrowed key, but nothing hides envelope information. This is seemingly a fundamental limitation: if a user's messages can easily be located (and subsequently decrypted with an escrowed key), then the privacy protections are severely lacking as such a system leaks metadata—i.e., who messages are going to and coming from. If it is impossible to locate a user's message given their name, locating the messages to decrypt relies on expensive operations such as trial decryption with every escrowed key. This does not scale.

The question thus is, can we meet the requirements for providing lawful access while still allowing cryptographic systems that prevent dragnet surveillance of either messages or metadata? We begin with three insights:

1. There can be no central decryption key, since we cannot trust anyone to hold it.
2. Even after raising suspicion, a user will likely interact with the communication system (e.g., by logging in again). If they were cautious enough not to do so, they would be using existing end-to-end secure schemes already.
3. We do not need to prevent the authorities from abusing their position. Merely detecting if they do it, serves as a deterrent for mass surveillance.

Markulf Kohlweiss: Microsoft Research, E-mail: markulf@microsoft.com
***Corresponding Author: Ian Miers:** The Johns Hopkins University, E-mail: imiers@cs.jhu.edu

We feel this approach in general is quite promising. In this paper, we apply it to a somewhat more theoretical problem: building a group signature like scheme—which we call an accountable tracing signature—where the group manager can, on a recurring basis, (i) choose which users' messages they have the ability to open and (ii) prove that they did not have the ability to open the messages of a given user. Similar to placing a GPS tracker on a car or wiretapping a phone, this only provides information going forwards. We then extend this to allow retroactive access and the escrow of keys, rather than just user identities.

Group signatures are a useful starting point with well studied cryptographic definitions. They allow anonymity revocation by a group manager while still providing privacy to everyone else. Moreover, they are readily extendable to key escrow: many group signature schemes are a signature scheme plus an escrow mechanism for the users' identity. Including a decryption key in addition to the identity is simple. Additionally, by starting with group signatures, we inherit the strong guarantees that allow the group manager to prove that someone authored a message if-and-only-if they actually did so, i.e., she cannot frame users. While this property is understandably neglected in many escrow schemes, given the legal context in which escrowed messages are likely to be used as evidence in court, attribution seems to be a fundamental property that should be dealt with. Finally, group signatures are third party verifiable, enabling an untrusted mail transport agent or telecom switch to verify that messages comply with required lawful access requests without itself having that access.

In group signatures, however the group manager can open any message and is simply trusted not to. For accountable escrow, this is not the case and the group manager is fundamentally an adversary who seeks to actively violate user privacy. We model this by adding two additional games. The first one—acountable anonymity—ensures that the group manager cannot both trace a user and provide proof that they did not do so. The second—trace obliviousness—ensures that users cannot tell if they are traceable or not (whereas in standard group signatures, the user knows they are always traced).

Our approach to constructing a scheme meeting these is as follows: as in a group signature, users escrow their identity under an escrow public key as part of the signature. Normally, however, their private key is not known to the authority. When a user resubscribes (e.g., pays their monthly phone bill or logs into a webmail account), the authority searches them by either (i) replacing that key with one the authority can decrypt, or (ii) requiring the user to provide the escrow private key encrypted under an authority provided key. The former results in tracing going forward—anything the user does going forward is detectable—while the latter allows backward tracing as well and reveals past actions. In case a user is not traced, the authority does not replace keys and merely re-randomizes the existing public key.

The core of this approach is a key-oblivious encryption scheme where public keys are randomizable and randomized keys cannot be linked to each other or the original key. As a result, without the entropy used in key randomization, users cannot tell if their key was replaced—as in the case of forward tracing—or if they are encrypting to a random group element or the authority's public key—as in the case of backward tracing. Because users cannot tell if either search mechanism was invoked, users remain oblivious. However, once the randomness is revealed, the authority is held accountable.

Our Contribution

We provide four contributions. First, we propose accountability as a novel security requirement of escrow systems and formalize it in the definitions for accountable tracing signatures. Second, we provide a practical construction for an accountable forward-tracing signature complete with proofs of security. This allows an authority to covertly and accountably tag and trace user messages once the user becomes suspect. Third, we extend our approach with an interactive subscribe protocol to build an accountable backward-tracing signature scheme where all of a user's messages can be identified even retroactively. Finally, we show how to augment either approach to create an accountable *tracing signcryption* scheme where messages are encrypted and opening a signature reveals both the author and the message content, giving us an efficient accountable wiretapping system.

Related work

Several cryptographic schemes, both from the academic literature and in practice, use a trusted party for escrow. For example, in a group signature scheme [8], a group manager allows users to sign messages as coming from some member of the group while he alone maintains the ability to provably identify who signed the message. A related problem is given a suspicious user, identify all messages they have authored. Systems that support both functionalities are called traceable signatures [14].

Variants of these properties have been defined for e-cash systems, with owner tracing, and coin tracing being defined analogously. Interestingly there are e-cash schemes that hold the authority accountable [15, 16]. Unfortunately, these schemes require interaction with the tracing authority for every transaction and, as a result, the techniques do not necessarily scale.

Finally, key escrow, where an encrypted message can be decrypted by both the recipient and an escrow authority, is both a well studied topic and one that has seen at least limited real world deployment in the (failed) Clipper chip. A key escrow mechanism in which multiple trustees need to collaborate to decrypt has been proposed by [18].

2 Key-oblivious encryption

We call a public-key encryption scheme *key-oblivious* if (i) it allows for a large set of public keys all related to the same secret key, if (ii) existing public keys can be randomized to generate related keys, and if (iii) it cannot be discerned, without knowledge of the secret key and the randomness used in their generation, how public keys are related. The existence of such schemes is cryptographic folklore and we do not claim much novelty here. In order to allow for accountability, we insist on the ability to prove, for a given public key, which key was randomized to produce it. This leads to slight variants of the standard key-privacy and plaintext-indistinguishability games.

2.1 Syntax

We formalize a key-oblivious encryption scheme OE as a collection $\mathsf{OE}.(\mathsf{GroupGen}, \mathsf{KeyGen}, \mathsf{KeyRand}, \mathsf{Enc}, \mathsf{Dec})$ of five algorithms:

$\mathsf{GroupGen}(1^\lambda) \to \mathcal{G}$**:** generates parameters, usually a prime order group.

$\mathsf{KeyGen}(\mathcal{G}) \to (pk, sk)$**:** generates a key pair.

$\mathsf{KeyRand}(pk) \to pk'$**:** randomizes an existing public key into a public key for the same secret key.

$\mathsf{Enc}(pk, m) \to ct$**:** standard encryption functionality.

$\mathsf{Dec}(sk, ct) \to m$**:** standard decryption functionality.

In definitions and in protocols, we sometimes make the randomness of $\mathsf{KeyRand}$ explicit and write $(pk'; r) \leftarrow \mathsf{KeyRand}(pk)$ and $pk' = \mathsf{KeyRand}(pk; r)$. For two fixed random public keys $pk^{(0)}, pk^{(1)}$, $pk = \mathsf{KeyRand}(pk^{(b)}; r)$ acts as a *hiding* and *binding* bit commitment scheme, with r its opening. (Similarly, we write $(ct; s) \leftarrow \mathsf{Enc}(pk, m)$ and $ct = \mathsf{Enc}(pk, m; s)$ to make the randomness of Enc explicit in zero-knowledge proofs.)

2.2 Definitions

In addition to *key randomizability*—which corresponds to the bit commitment being hiding, we require that ciphertexts are plaintext indistinguishable even when the adversary can randomize the target keys. We term this *plaintext indistinguishability under key randomization* (INDr). We also require such schemes to be *key private* [5], again with the modification that this holds even for adversarially randomized keys. Even though stronger variants of these properties exist, security under chosen plaintext attacks suffices for our purposes. We note that any key randomizable encryption scheme can be made key private via the added step of randomizing the public key prior to encryption.

Game $\mathsf{KR} \triangleq$
$b \leftarrow \{0, 1\}$
$\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\mathcal{G})$
$pk_0 \leftarrow \mathsf{KeyRand}(pk)$
$(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(\mathcal{G})$
$b' \leftarrow \mathcal{A}(pk, pk_b)$
return $(b' = b)$

Game $\mathsf{INDr} \triangleq$
$b \leftarrow \{0, 1\}$
$\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\mathcal{G})$
$(pk', r, m_0, m_1, st) \leftarrow \mathcal{A}_0(\mathcal{G}, pk)$
if $pk' \neq \mathsf{KeyRand}(pk; r)$ **then**
  return $\perp$
$ct \leftarrow \mathsf{Enc}(pk', m_b)$
$b' \leftarrow \mathcal{A}_1(ct, st)$
return $(b' = b)$

Game $\mathsf{KPr} \triangleq$
$b \leftarrow \{0, 1\}$
$\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$
$(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(\mathcal{G})$
$(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(\mathcal{G})$
$(m, pk'_0, r_0, pk'_1, r_1, st)$
  $\leftarrow \mathcal{A}_0(\mathcal{G}, pk_0, pk_1)$
if $\neg(pk'_i = \mathsf{KeyRand}(pk_i; r_i)$
  for $i \in \{0, 1\})$ **then**
  return $\perp$
$ct \leftarrow \mathsf{Enc}(pk'_b, m)$
$b' \leftarrow \mathcal{A}_1(ct, st)$
return $(b' = b)$

Definition 1 (Key randomizability)**.** *Let* OE *be a key-oblivious encryption scheme. Consider Game* KR *played by adversary* $\mathcal{A}$*: The key-randomizability advantage of* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathsf{KR}}(\mathcal{A})$ *is defined as* $2 \cdot Pr[\mathsf{KR} : \mathsf{true}] - 1$*. A scheme is key randomizable if for any polynomial time* $\mathcal{A}$ *this advantage is negligible.*

Definition 2 (Key privacy under key randomization). *Let* OE *be a key-oblivious encryption scheme. Consider Game* KPr *played by adversary* $\mathcal{A}$*: The key-privacy advantage of* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathsf{KPr}}(\mathcal{A})$ *is defined as* $2 \cdot Pr[\mathsf{KPr} : \mathsf{true}] - 1$*. A scheme is key private if for any polynomial time* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *this advantage is negligible.*

Definition 3 (Plaintext indistinguishability under k.r). *Let* OE *be a key-oblivious encryption scheme. Consider Game* INDr *played by adversary* $\mathcal{A}$*: The plaintext distinguishing advantage of* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathsf{INDr}}(\mathcal{A})$ *is defined as* $2 \cdot Pr[\mathsf{INDr} : \mathsf{true}] - 1$*. A scheme is secure if for any polynomial time* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *this advantage is negligible.*

2.3 Instantiation

Intuitively key-oblivious encryption can be obtained from key-private homomorphic public-key encryption. For a given key pair, let $C(m; r)$ be the ciphertext of message m with randomness r. Consider the homomorphic property $C(m; r) \otimes C(m'; r') = C(m \cdot m'; r + r')$. We construct a key-oblivious encryption scheme by letting each oblivious public key corresponds to an encryption $C(1; r)$. We randomize a key by computing $C(1; r * r')$ for a random r' (this can be computed using $\otimes$ by square-and-multiply). Encryption corresponds to computing $C(m; r+s)$ by multiplying a freshly randomized public key $C(1; r + s)$ with the ciphertext $C(m; 0)$, i.e., the message encrypted with randomness zero—see also [11].

We will use such a simple construction of key-oblivious encryption based on ElGamal (ElKO):

- ElKO.GroupGen(1^λ)**:** Pick a group $\mathbb{G}$ of prime order q with generator g, and return $\mathcal{G} = (\mathbb{G}, q, g)$.
- ElKO.KeyGen($\mathcal{G}$)**:** Sample $r, \gamma \leftarrow \mathbb{Z}_q$ and return $pk = (g^r, (g^\gamma)^r)$, $sk = (pk, \gamma)$.
- ElKO.KeyRand(pk)**:** Let $pk = (A, B)$, sample $r \leftarrow \mathbb{Z}_q$, return $pk' = (A^r, B^r)$ and randomness r.
- ElKO.Enc(pk, m)**:** Let $pk = (A, B)$, sample $s \leftarrow \mathbb{Z}_q$, and return $ct = (A^s, B^s \cdot m)$.
- ElKO.Dec(sk, ct)**:** Let $sk = (pk, \gamma)$ and $ct = (C, D)$. Return $m = D/C^\gamma$.

Lemma 4 (Key privacy of ElKO). *If the decisional Diffie-Hellman problem holds in* $\mathbb{G}$*,* ElKO *is a key-private key-oblivious encryption scheme.*

Lemma 5 (Key randomizability of ElKO). *If the decisional Diffie-Hellman problem holds in* $\mathbb{G}$*,* ElKO *is a key-randomizable encryption scheme.*

Lemma 6 (Plaintext indistinguishability of ElKO). *If the decisional Diffie-Hellman problem holds in* $\mathbb{G}$*,* ElKO *satisfies* INDr*.*

2.3.1 SIG and OTS

To ensure efficiency, we need structure-preserving signatures (SPS) [4]. SPS can either be proved secure in the generic group model [3], using q-type assumptions [4], or using static assumptions [1]. Usually the stronger the assumption, the better the performance. This gives us several options. Moreover, if we only use the signature scheme to sign freshly created one-time signature (OTS) public keys, we do not require full adaptive unforgeability, but it suffices if the signature scheme is secure for random messages. We can thus use the xSIG scheme from [1] secure under DDH_2 and $xDLIN_1$ with $|\sigma_{sig}| = 6$. For the OTS scheme, which does not need to be structure preserving, on the other hand, we will always make use of the scheme of [13] with $|pk_{ots}| = 2$ and $|\sigma_{ots}| = 2$. Note that the xSIG scheme requires random messages of a specific form, and thus requires extended pk_{ots} of size $2 \times 3 = 6$.

3 Accountable forward-traceable signatures

In a group signature scheme, a group manager can open any suspect message. In a tracing signature scheme, the manager can test if any message belongs to a suspect user. In an accountable tracing signature, the manager can do the same but must later reveal which users she deems suspect.

3.1 Syntax

We formalize an accountable tracing signature scheme ATS as a collection ATS.(Setup, GKg, UKg, Enroll, Sign, Verify, Open, Judge, Account) of nine algorithms:

- Setup(1^λ) $\rightarrow gp$**:** Generates the public parameters for security level λ.
- GKg(gp) $\rightarrow (gpk, gsk)$**:** Generates the initial group key pair.

$\mathsf{UKg}(gp) \rightarrow (upk, usk)$**:** Generates a user key pair. Each user has a public key *upk* and a corresponding private key *usk*. Per [6] this is necessary to provide any meaning to the assertion that a user actually did sign an opened message: without it, the group manager is free to simply assert that a key of their generation actually belongs to a user.

$\mathsf{Enroll}(gsk, upk, epoch, t) \rightarrow (cert, w^{escrw})$**:** The authority produces a certificate on a user's escrow public key. This certificate either provides full anonymity ($t = 0$) or allows for tracing ($t = 1$) depending on the bit t. The authority stores the witness w^{escrw} to t and returns the certificate to the user. The certificate *cert* contains the time range for which the user is enrolled. We call this counter the *epoch* of the certificate.

$\mathsf{Sign}(gpk, cert, usk, m) \rightarrow \sigma$: Takes the group public key, a certificate, the user's private key, and a message to sign as input. It outputs a signature that may contain an escrow of the user's identity. The *epoch* of the signature corresponds to the *epoch* of the certificate.

$\mathsf{Verify}(gpk, m, \sigma, epoch) \rightarrow \{0, 1\}$**:** Given the group public key, a message, its signature, and an epoch, verifies the signature is valid for the specified message and epoch.

$\mathsf{Open}(gpk, gsk, m, \sigma, epoch) \rightarrow (upk, \psi)$**:** Given the group public key, private key, a message, its signature, and the time interval; if possible (i.e., if signed using an escrow certificate), return the public key of the user and a proof that the user generated the message-signature pair. Otherwise returns $\perp$.

$\mathsf{Judge}(gpk, m, \sigma, epoch, upk, \psi) \rightarrow \{0, 1\}$**:** Given the group public key, a message, its signature, an epoch, a user's public key, and a proof that the user generated the signature of that epoch on that message, verifies the proof.

$\mathsf{Account}(gpk, cert, w^{escrw}, t) \rightarrow \{0, 1\}$**:** Given the group public key, a certificate, a bit t saying whether it escrows the user's identity, and a witness w^{escrw}, returns 1 if the witness confirms the choice of t.

We will expose some implementation details for the sake of simplifying definitions and avoiding excessive scaffolding. We use $cert.epoch$ to denote the epoch of a certificate, $cert.upk$ to denote the user public key being certified, $gpk.gp$ the parameters of the group public key, and $gsk.csk$ to denote the certificate signing key—this makes an adversary with access to this key strictly stronger than an adversary with access to an ENROLL oracle.

3.2 Definitions

The games defining the properties of an accountable tracing signature are described formally in Figure 1. We outline them informally here. The first three are just slight adaptations of the standard group signature games.

The last two, *anonymity with accountability* and *trace-obliviousness*, are fundamentally different from the guarantees that group signatures can provide. These stem from the requirement to hold the group manager accountable.

Anonymity under tracing This corresponds to the standard anonymity property of group signatures which guarantees anonymity toward everyone except the group manager. It ensures that even when being traced users are anonymous to the general public.

Traceability Informally, traceability requires that every valid message will trace to someone as long as the adversary does not have a certificate and private key for a non-tracing certificate.
Although the attacker is free to choose the type of certificate for honest users whose keys are generated by the UKG oracle, he is not allowed to get untraceable certificates on user keys of his choosing. In the definition this is ensured by $(t \vee upk \notin \mathsf{dom}(S))$ always evaluating to 1 on such occasions. This is a slight departure from the standard tracing game [6], where ENROLL always produces traceable certificates.

Non-frameability This requires that no one, not even the group manager, can sign messages on the user's behalf. At the same time, it guarantees that users, if they are being traced, have to take responsibility for the messages they sign, i.e., traced signatures ensure non-repudiation.

Anonymity with accountability Intuitively, this captures the notion that the user is anonymous even from a corrupt authority that has full control of the system. It requires that even if every single parameter in the system is adversarially controlled, a user is anonymous as long as the escrow key in his certificate is accountably private.

Trace-obliviousness This requires that users cannot tell whether they are being traced.

Definition 7 (Anonymity under tracing)**.** *Let* ATS *be an accountable forward-tracing signature scheme. Consider the following game played by an adversary $\mathcal{A}$:*

$$
\begin{array}{l}
\textbf{Game } \mathsf{AuT} \triangleq\\
\quad b \leftarrow \{0,1\}\\
\quad gp \leftarrow \mathsf{Setup}(1^\lambda)\\
\quad (gpk, gsk) \leftarrow \mathsf{GKg}(gp)\\
\quad b' \leftarrow \mathcal{A}^{\mathsf{Ch},\mathsf{OPEN}}(gpk, gsk.csk)\\
\quad \textbf{return } (b' = b)
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{Ch}(sk_0, sk_1, cert_0, cert_1, m, w_0^{escrw}, w_1^{escrw}, t) \triangleq\\
\quad \sigma_0 \leftarrow \mathsf{Sign}(gpk, sk_0, cert_0, m)\\
\quad \sigma_1 \leftarrow \mathsf{Sign}(gpk, sk_1, cert_1, m)\\
\quad \textbf{if } (\sigma_0 \neq \bot \wedge \sigma_1 \neq \bot \wedge cert_0.epoch = cert_1.epoch \wedge\\
\qquad \mathsf{Account}(gpk, cert_0, w_0^{escrw}, t) \wedge \mathsf{Account}(gpk, cert_1, w_1^{escrw}, t))\\
\qquad Q \leftarrow Q \cup \{\sigma_b\}\\
\qquad \textbf{return } \sigma_b\\
\quad \textbf{else}\\
\qquad \textbf{return } \bot
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{OPEN}(m, \sigma) \triangleq\\
\quad \textbf{if } (\sigma \in Q) \textbf{ then return } \bot\\
\quad \textbf{return } \mathsf{Open}(gsk, m, \sigma)
\end{array}
$$

The anonymity under tracing advantage of $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{AuT}}(\mathcal{A})$ is defined as $2 \cdot Pr[\mathsf{AuT} : \mathsf{true}] - 1$. ATS *is anonymous under tracing if for any polynomial time $\mathcal{A}$ this advantage is negligible.*

Definition 8 (Traceability)**.** *Let* ATS *be an accountable forward-tracing signature scheme. Consider the following game played by adversary $\mathcal{A}$:*

$$
\begin{array}{l}
\textbf{Game } \mathsf{Trace} \triangleq\\
\quad gp \leftarrow \mathsf{Setup}(1^\lambda)\\
\quad (gpk, gsk) \leftarrow \mathsf{GKg}(gp)\\
\quad (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{UKG},\mathsf{ENROLL},\mathsf{SIGN},\mathsf{OPEN}}(gpk)\\
\quad (upk, \psi) \leftarrow \mathsf{Open}(gsk, m, \sigma)\\
\quad \textbf{return } (\ \mathsf{Verify}(gpk, m, \sigma) = 1 \wedge\\
\qquad (m, \sigma) \notin Q \wedge\\
\qquad (\mathsf{Judge}(gpk, m, \sigma, upk, \psi) = 0 \vee\\
\qquad\ upk = \bot)\ )
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{UKG}(upk, epoch) \triangleq\\
\quad (upk, usk) \leftarrow \mathsf{UKg}(gp)\\
\quad S[upk] \leftarrow usk\\
\quad \textbf{return } upk
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{ENROLL}(upk, epoch, t) \triangleq\\
\quad (cert, w^{escrw})\\
\qquad \leftarrow \mathsf{Enroll}(gsk, upk, epoch,\\
\qquad\quad (t \vee upk \notin \mathsf{dom}(S)))\\
\quad \textbf{return } cert
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{OPEN}(m, \sigma) \triangleq\\
\quad (upk, \psi) \leftarrow \mathsf{Open}(gsk, m, \sigma)\\
\quad \textbf{return } (upk, \psi)
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{SIGN}(cert, m) \triangleq\\
\quad usk = S[cert.upk]\\
\quad \textbf{if } (usk = \bot) \textbf{ then return } \bot\\
\quad \sigma \leftarrow \mathsf{Sign}(gpk, cert, usk, m)\\
\quad Q \leftarrow Q \cup \{(m, \sigma)\};\ \textbf{return } \sigma
\end{array}
$$

The traceability advantage of $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{Trace}}(\mathcal{A})$ is defined as $Pr[\mathsf{Trace} : \mathsf{true}]$. ATS *is traceable if for any polynomial time $\mathcal{A}$ this advantage is negligible.*

Definition 9 (Non-frameability)**.** *Let* ATS *be an accountable forward-tracing signature scheme. Consider the following game played by adversary $\mathcal{A}$:*

$$
\begin{array}{l}
\textbf{Game } \mathsf{NF} \triangleq\\
\quad gp \leftarrow \mathsf{Setup}(1^\lambda)\\
\quad (gpk, st) \leftarrow \mathcal{A}_0(gp)\\
\quad \textbf{if } gpk.gp \neq gp \textbf{ then}\\
\qquad \textbf{return } \bot\\
\quad (m, \sigma, upk, \psi) \leftarrow \mathcal{A}_1^{\mathsf{UKG},\mathsf{SIGN}}(st)\\
\quad \textbf{return } (\mathsf{Judge}(gpk, m, \sigma, upk, \psi) = 1\\
\qquad \wedge (m, \sigma) \notin Q \wedge upk \in S
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{UKG}(upk, epoch) \triangleq\\
\quad (upk, usk) \leftarrow \mathsf{UKg}(gp)\\
\quad S[upk] \leftarrow usk\\
\quad \textbf{return } upk
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{SIGN}(cert, m) \triangleq\\
\quad usk = S[cert.upk]\\
\quad \textbf{if } (usk = \bot) \textbf{ then return } \bot\\
\quad \sigma \leftarrow \mathsf{Sign}(gpk, cert, usk, m)\\
\quad Q \leftarrow Q \cup \{(m, \sigma)\};\ \textbf{return } \sigma
\end{array}
$$

The non-frameability advantage of $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{NF}}(\mathcal{A})$ is defined as $Pr[\mathsf{NF} : \mathsf{true}]$. ATS *is non-frameable if for any polynomial time $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ this advantage is negligible.*

Definition 10 (Anonymity with accountability)**.** *Let* ATS *be an accountable forward-tracing signature scheme. Consider the following game played by an adversary $\mathcal{A}$:*

$$
\begin{array}{l}
\textbf{Game } \mathsf{AwA} \triangleq\\
\quad b \leftarrow \{0,1\}\\
\quad gp \leftarrow \mathsf{Setup}(1^\lambda)\\
\quad (gpk, st) \leftarrow \mathcal{A}_0(gp)\\
\quad \textbf{if } gpk.gp \neq gp\\
\qquad \textbf{return } \bot\\
\quad b' \leftarrow \mathcal{A}_1^{\mathsf{Ch}}(st)\\
\quad \textbf{return } (b' = b\ \ )
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{Ch}(sk_0, sk_1, cert_0, cert_1, m, w_0^{escrw}, w_1^{escrw})) \triangleq\\
\quad \sigma_0 = \mathsf{Sign}(gpk, sk_0, cert_0, m)\\
\quad \sigma_1 = \mathsf{Sign}(gpk, sk_1, cert_1, m)\\
\quad \textbf{if } (\sigma_0 \neq \bot \wedge \sigma_1 \neq \bot \wedge cert_0.epoch = cert_1.epoch \wedge\\
\qquad \mathsf{Account}(gpk, cert_0, w_0^{escrw}, 0) \wedge \mathsf{Account}(gpk, cert_1, w_1^{escrw}, 0))\\
\qquad \textbf{return } \sigma_b\\
\quad \textbf{else}\\
\qquad \textbf{return } \bot
\end{array}
$$

The anonymity advantage of $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{AwA}}(\mathcal{A})$ is defined as $2 \cdot Pr[\mathsf{AwA} : \mathsf{true}] - 1$. ATS *is anonymous if for any polynomial time $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ this advantage is negligible.*

Definition 11 (Trace-obliviousness)**.** *Let* ATS *be an accountable forward-tracing signature scheme. Consider the following game played by an adversary $\mathcal{A}$:*

$$
\begin{array}{l}
\textbf{Game } \mathsf{TO} \triangleq\\
\quad gp \leftarrow \mathsf{Setup}(1^\lambda)\\
\quad (gpk, gsk) \leftarrow \mathsf{GKg}(gp)\\
\quad b \leftarrow \{0,1\}\\
\quad (b') \leftarrow \mathcal{A}^{\mathsf{Ch},\mathsf{ENROLL},\mathsf{OPEN}}(gpk)\\
\quad \textbf{return } (b' = b\ \ )
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{OPEN}(m, \sigma) \triangleq\\
\quad (upk, \psi) \leftarrow \mathsf{Open}(gsk, m, \sigma)\\
\quad \textbf{if } upk \in U \textbf{ then}\\
\qquad \textbf{return } \bot\\
\quad \textbf{else}\\
\qquad \textbf{return } (upk, \psi)
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{ENROLL}(upk, epoch, t) \triangleq\\
\quad (cert, w^{escrw}) \leftarrow \mathsf{Enroll}(gsk, upk, epoch, t)\\
\quad \textbf{return } cert
\end{array}
$$

$$
\begin{array}{l}
\textbf{Oracle } \mathsf{Ch}(upk, epoch) \triangleq\\
\quad (cert, w^{escrw}) \leftarrow \mathsf{Enroll}(gsk, upk, epoch, b)\\
\quad U = U \cup \{upk\};\ \textbf{return } cert
\end{array}
$$

The trace-obliviousness advantage of $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{TO}}(\mathcal{A})$ is defined as $2 \cdot Pr[\mathsf{TO} : \mathsf{true}] - 1$. ATS *is trace oblivious if for any polynomial time $\mathcal{A}$ this advantage is negligible.*

Fig. 1. Definitions

3.2.1 On the necessity of extra games

Although at first glance anonymity with accountability is similar to the standard anonymity property and may appear to subsumes it, this is not the case: even if the authority can lawfully trace a user, we still need to ensure that no one else can.

From this, it appears that trace-obliviousness together with anonymity with accountability subsumes the standard anonymity property. After all, if a third party can track a traced user, then surely the user can use the same technique to detect their own status.

If anonymity under tracing were limited to generic third parties, this might be the case. However, in both standard group signature schemes and ours, it is useful to allow the opening authority to be distinct from the group manager for security reasons: the former is used infrequently while the latter is online continually to allow new members in. Ideally, compromise of the online portion ought not to violate users' privacy, and so the attacker in the anonymity under tracing game may get access to the online portion's state ($gsk.csk$). Trace-obliviousness, however, cannot necessarily survive such a strong attack, as the group manager may know whose keys have been replaced. As a result, the games are distinct.

3.2.2 Remarks.

Note two things. First, if all keys are tracing keys, then (i) the trace game would be standard, (ii) the trace-oblivious game is moot, and (iii) there is no accountable anonymity. As a result, we would have a standard group signature scheme. Perhaps more interestingly, if we dropped the requirement for anonymity under tracing, we would not require simulation extractability for the proof system.

Note, moreover, that the anonymity under tracing adversary $\mathcal{A}$ provides all input to Ch. A simple hybrid argument thus shows that anonymity under tracing with a single challenge query implies security with multiple challenge queries. This means that even one-time simulation extraction is sufficient to prove security.

4 Accountable forward-tracing signatures from key-oblivious encryption

Assume we have an unforgeable signature scheme $\mathsf{SIG}.(\mathsf{GroupGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, a one-time signature scheme $\mathsf{OTS}.(\mathsf{GroupGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, a key-oblivious encryption scheme $\mathsf{OE}.(\mathsf{GroupGen}, \mathsf{KeyGen}, \mathsf{KeyRand}, \mathsf{Enc}, \mathsf{Dec})$, and a simulation-extractable non-interactive zero-knowledge proof system $\Pi.(\mathsf{GroupGen}, \mathsf{Setup}, \mathsf{Prove}, \mathsf{VfyProof}, \mathsf{SimExtSetup}, \mathsf{Sim}, \mathsf{Ext})$. (For efficiency we require that $\mathsf{SIG.GroupGen} = \mathsf{OTS.GroupGen} = \mathsf{OE.GroupGen} = \Pi.\mathsf{GroupGen}$ and refer to this algorithm as $\mathsf{GroupGen}$.)

We construct an accountable tracing scheme ATS:

$\mathsf{ATS.Setup}(1^\lambda)$: Runs $\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$, $(pk^{(0)}, sk^{(0)}) \leftarrow \mathsf{OE.KeyGen}(\mathcal{G})$, $crs \leftarrow \Pi.\mathsf{Setup}(\mathcal{G})$. The secret key $sk^{(0)}$ is discarded, in fact $pk^{(0)}$ should be generated in such a way that $sk^{(0)}$ is not known to any party. Outputs $gp = (\mathcal{G}, pk^{(0)}, crs)$.

$\mathsf{ATS.GKg}(gp)$: Runs $(cpk, csk) \leftarrow \mathsf{SIG.Keygen}(gp.\mathcal{G})$ and $(pk^{(1)}, sk^{(1)}) \leftarrow \mathsf{OE.KeyGen}(gp.\mathcal{G})$. Returns $gpk = (gp, cpk, opk = pk^{(1)})$ and $gsk = (gpk, csk, osk = sk^{(1)})$.

$\mathsf{ATS.UKg}(gp)$: Returns $(upk, usk) \leftarrow \mathsf{SIG.Keygen}(gp.\mathcal{G})$.

$\mathsf{ATS.Enroll}(gsk, upk, epoch, b)$: Given a user public key, computes $(epk; w^{escrw}) \leftarrow \mathsf{OE.KeyRand}(pk^{(b)})$ and a signature $\sigma_{cert} \leftarrow \mathsf{SIG.Sign}(csk, (upk, epk, epoch))$. Returns certificate $cert = ((upk, epk, epoch), \sigma_{cert})$ and witness w^{escrw}.

$\mathsf{ATS.Sign}(gpk, cert, usk, m)$: Parses gpk as (gp, cpk, opk) and $cert$ as $((upk, epk, epoch), \sigma_{cert})$. Runs $(pk_{ots}, sk_{ots}) \leftarrow \mathsf{OTS.Keygen}(gp)$, $\sigma_u \leftarrow \mathsf{SIG.Sign}(usk, pk_{ots})$, and $(escrw; s) \leftarrow \mathsf{OE.Enc}(epk, (upk, \sigma_u))$. Computes a proof π using crs for the following relation to prove knowledge of $(upk, epk, \sigma_{cert}, \sigma_u)$:

$$
\begin{aligned}
&((escrw, pk_{ots}, cpk, epoch), (upk, epk, \sigma_{cert}, \sigma_u, s)) \in R_{sig} \\
&\text{iff } (\mathsf{SIG.Verify}(cpk, \sigma_{cert}, (upk, epk, epoch)) = 1 \wedge \\
&\qquad escrw = \mathsf{OE.Enc}(epk, (upk, \sigma_u); s) \wedge \\
&\qquad\qquad \mathsf{SIG.Verify}(upk, \sigma_u, pk_{ots}) = 1)\ .
\end{aligned}
$$

Computes $\sigma_{ots} \leftarrow \mathsf{OTS.Sign}(sk_{ots}, (m, (escrw, pk_{ots}, cpk, epoch), \pi))$ and sets $\sigma = (\pi, \sigma_{ots}, pk_{ots}, escrw)$. If $\mathsf{ATS.Verify}(gpk, m, \sigma) = 0$, returns $\perp$, otherwise returns σ. This check is needed as we guarantee anonymity even for maliciously formed inputs, as long as the signature verifies and the user is not being traced.

ATS.Verify($gpk, m, \sigma, epoch$): Parses gpk as (gp, cpk, opk) and σ as $(\pi, \sigma_{ots}, pk_{ots}, escrw)$. First, verifies the one-time signature: OTS.Verify($pk_{ots}, \sigma_{ots}, (m, (escrw, pk_{ots}, cpk, epoch), \pi)$). Second, verifies the proof Π.VfyProof($crs, (escrw, pk_{ots}, cpk, epoch), \pi$). If all checks succeed, returns 1, otherwise returns 0.

ATS.Open($gpk, gsk, m, \sigma, epoch$): Parses gpk as (gp, cpk, opk) and σ as $(\pi, \sigma_{ots}, pk_{ots}, escrw)$. Runs ATS.Verify($gpk, m, \sigma, escrw$) and abort if it fails. Extracts osk from gsk and runs $(upk, \sigma_u) =$ OE.Dec($osk, escrw$). Returns $(upk, \psi = \sigma_u)$.

ATS.Judge($gpk, m, \sigma, epoch, upk, \psi$): Parses gpk as (gp, cpk, opk), σ as $(\pi, \sigma_{ots}, pk_{ots}, escrw)$, and ψ as σ_u. Runs ATS.Verify($gpk, m, \sigma, epoch$) and then runs SIG.Verify(upk, σ_u, pk_{ots}). If both checks succeed, returns 1, otherwise 0.

ATS.Account($gpk, cert, w^{escrw}, b$): Returns 1, if $cert.epk =$ OE.KeyRand($pk^{(b)}; w^{escrw}$), otherwise 0.

Theorem 12. *If* SIG *is unforgeable,* OTS *is strongly unforgeable,* OE *is plaintext indistinguishable, key private and key randomizable, and* Π *is zero-knowledge and simulation-extractable then* ATS *is anonymous under tracing, traceable, non-frameable, accountably anonymous, and trace oblivious as defined in Section 3.2.* See the proofs of Theorems 13-21 in Appendix B.1.

We detail the concrete costs of instantiating such a scheme in Appendix A. Depending on techniques, a signature in our scheme requires between 155 and 367 group elements. Depending on the type of curve, group elements are between 32 and 128 bytes each for BN256 Curves with 128 bit security, this gives us a signature between 11Kb and 45Kb. This makes our scheme of mostly theoretical interest as a proof of concept.

We note, however, that there may be substantial room for improvement. First, our construction simply uses stock parts and it's possible that a bespoke solution would give far better performance. Second, we use structure preserving signatures and Groth-Sahai proofs. It's possible that if we draw our stock parts from signatures with efficient protocols and verifiable encryption schemes, we will get a more efficient scheme. Similarly, schemes secure in the Random-Oracle model generally have more efficient protocols and our current construction is in the more expensive standard-model.

5 Backward-tracing and message-escrow extensions

We only formally describe and analyze a base scheme, though our approach can be extended in several directions to fit specific application requirements. We discuss two such extensions for applications that require backward-tracing and encryption respectively.

5.1 Accountable backward-tracing signatures

So far we have considered monitoring a suspect's future actions. In the case of recovering past actions, we cannot retroactively tag a message and must, instead, extract something from the user to identify her messages.

With some applications (e.g., cloud based email), where users may maintain an encrypted inbox/outbox of their messages merely (accountably) extracting the necessary decryption key is sufficient. We can decrypt the inbox rather than resort to trial message decryption. For other applications, it seems search costs are on the order of $m \times t$ where m is the number of messages in the system and t is the number of targeted users.

In either case, retrieving the user's key introduces a second issue: restoring privacy. For forward tracing, the authority merely needs to replace the escrow key with a randomization of $pk^{(0)}$ when a warrant expires. For backward tracing, things are more complicated as we need the user to replace her key with a new one without realizing she did so. This requires more than just key obliviousness: the user must only hold a share of her private key. If not, she can simply test if she can decrypt messages encrypted under the new key.

We augment EIKO with a basic distributed key generation functionality to form DEIKO. We model distributed key generation using the following algorithms: ShareGen($\mathcal{G}$) generates public and private key shares (ps, ss), while CombinePS and CombineSS combine a vector of public and private key shares into a public and a private key respectively. We extend EIKO with algorithms ShareGen, CombinePS and CombineSS for generating keys from public and private shares:

DEIKO.ShareGen($\mathcal{G}$): $\alpha \leftarrow \mathbb{Z}_q$, $ps = g^{\alpha}$, $ss \leftarrow \alpha$. Returns (ps, ss).

DEIKO.CombinePS($\vec{ps}$): $r \leftarrow \mathbb{Z}_q$, Returns $pk = (g^r, (\prod ps_i)^r)$.

DEIKO.CombineSS($\vec{ss}$): Returns $sk = \sum ss_i$.

5.1.1 Adding backward tracing

An Accountable backward-tracing signature scheme is a set of eight algorithms: $(\mathsf{Setup}, \mathsf{GKg}, \mathsf{UKg}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open}, \mathsf{Judge}, \mathsf{Account})$ and one protocol $\mathsf{Subscribe}(gpk, usk) \leftrightarrow \mathsf{Enroll}(gsk, upk, b)$.

Our construction is based on our forward-tracing scheme from Section 4 with two modifications (i) *epk* instead of being a key for EIKO is now for DEIKO and (ii) we replace Enroll with an interactive protocol (Subscribe, Enroll), that handles key generation, key retrieval in the case of a warrant, and key renewal on warrant expiration.

Subscribe, detailed in Figure 2, uses a blind decryption scheme [12] with algorithms $\mathsf{BE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Blind}, \mathsf{BlindDec}, \mathsf{UnBlind})$. Intuitively, in Subscribe, the user provides the authority with an encryption of her share of the escrow key. The authority can gain oblivious access to this share via a blind decryption query. To maintain trace-obliviousness, the authority normally issues blind decryption queries on an encryption of 0. Again, revealing the randomness—in this case for blinding the ciphertext—renders this accountable.

The process for key renewal is best understood via Figure 2. It leverages the fact that a user, since she knows only shares of escrow keys, cannot tell at the end of subscription whether she holds an escrow key generated from an old share, a new share, or a randomized key shared with all traced users.

5.2 Extending to encryption and message escrow

Both signature schemes can be readily adapted to form an escrowed encryption scheme by having the message be a ciphertext and including in *escrw* a copy of the plaintext, and modifying the proof in the signature accordingly. Formally, such as scheme has ten algorithms: $(\mathsf{Setup}, \mathsf{GKg}, \mathsf{UKg}, \mathsf{Enroll}, \mathsf{Signcrypt}, \mathsf{Decrypt}, \mathsf{Verify}, \mathsf{Open}, \mathsf{Judge}, \mathsf{Account})$. Sign is replaced by Signcrypt and augmented to additionally take a public key under which the message is encrypted. These keys can either be from an external source or produced by UKg. The resulting “signature” can be decrypted by Decrypt only with the corresponding private key. Verify can still be run by anyone of course.

Concretely, ATS.Sign becomes
$(\sigma, escrw) \leftarrow \mathsf{Signcrypt}(gpk, cert, usk, m, pk)$:
Parse gpk as (cpk, crs, opk) and
$cert$ as $(upk, epk, epoch)$.
Compute $(ct; s_{pk}) \leftarrow \mathsf{PKEnc}(pk, m)$.
Run $(pk_{ots}, sk_{ots}) \leftarrow \mathsf{OTS.Keygen}(gp)$,
$\sigma_u \leftarrow \mathsf{SIG.Sign}(usk, pk_{ots})$, and
$(escrw, s_{oe}) \leftarrow \mathsf{OE.Enc}(epk, (upk, \sigma_u, m))$.
Compute a proof π for the relation

$$\begin{aligned}
&(escrw, pk_{ots}, cpk, epoch, ct),\\
&\qquad (upk, \sigma_{cert}, \sigma_u, m, s_{oe}, s_{pk})) \in R_{sig}\\
&\text{iff } (\mathsf{SIG.Verify}(cpk, \sigma_{cert}, (upk, epk, epoch)) = 1 \wedge\\
&\qquad\qquad escrw = \mathsf{OE.Enc}(epk, (upk, \sigma_u, m); s_{oe}) \wedge\\
&\qquad\qquad ct = \mathsf{PKEnc}(pk, m; s_{pk}) \wedge\\
&\qquad\qquad \mathsf{SIG.Verify}(upk, \sigma_u, pk_{ots}) = 1)\ .
\end{aligned}$$

Run $\sigma_{ots} \leftarrow \mathsf{OTS.Sign}(sk_{ots}, (escrw, pk_{ots}, cpk, epoch, ct, \pi))$, set $\sigma = (\pi, \sigma_{ots}, pk_{ots}, escrw, ct)$ and if $\mathsf{ATS.Verify}(gpk, m, \sigma) = 0$, return $\perp$ else, return σ.

6 Transparency reports and conclusions

Accountable tracing signatures hold those who demand “lawful” access to encrypted messages accountable for what they access. With it, under some circumstances at least, demands for lawful access to cryptographic systems can be dealt with without allowing mass surveillance of message content and, crucially, metadata.

For most ordinary criminal cases, the existence of a search warrant is already revealed when data obtained from it is used in court. Thus the requirement to reveal searches after the fact is innocuous. However, many of the more troubling issues stem from orders which demand both access and silence. Currently, many companies issue transparency reports purportedly giving statistical data about the volume of such requests. These, again, are not accountable. Using an ATS scheme or its message escrow variant, however, these transparency reports become provable. If every *epk* is stored in a public ledger, then the authority can easily issue zero-knowledge proofs (e.g., using an efficient instantiation of zkSNARKS [19]) attesting that less than some fraction of its transactions use tracing keys for warrants accompanied by gag orders.

Accountable tracing signatures do not, of course, preclude the use of backdoors, software vulnerabilities, or other non-cryptographic attack vectors. However, given that they provide a vector for lawful access (and arguably bounded “unlawful” access to whatever extent the transparency report allows), they eliminate part of the motivation. Moreover, by potentially eliminating the

Subscribe(gpk, usk)		**Enroll**$(gsk, upk, epoch, b)$

		read$(upk, (epk_{old}, gps_{old}, gss_{old}, E_{old}, b_{old}, witnesses, keys))$
		$starttracing = b \wedge \neg b_{old}$;
		$stoptracing = b_{old} \wedge \neg b$
		$w_0 \leftarrow \$$
		if $(starttracing)$ **then**
		  $B =$ **BE.Blind**(E_{old}, w_0)
		else
		  $B =$ **BE.Blind**(E_0, w_0)

	$\xleftarrow{B}$	
$(ps, ss) \leftarrow$ **OE.ShareGen**(gp) $E \leftarrow$ **BE.Enc**(bpk, ss) $D =$ **BE.Dec**(bsk, B) π **is a proof of correct decryption**	$\xrightarrow{ps, E, D, \pi}$	

		$w_1 \leftarrow \$$
		verify the proof of correct decryption
		if $(starttracing)$
		  $ss_{old} =$ **BE.Unblind**(D, w_0)
		  $k =$ **OE.Combine**$SS(gss_{old}, ss_{old})$
		  $epk =$ **OE.KeyRand**$(pk^{(1)}; w_1)$
		  store$(upk, (epk, \perp, \perp, \perp, 1, witnesses, keys :: k))$
		else if $(stoptracing)$ **then**
		  $(gps, gss) \leftarrow$ **OE.ShareGen**(gp)
		  $epk =$ **OE.CombinePS**$(gps, ps; w_1)$
		  store$(upk, (epk, gps, gss, E, 0, witnesses :: (w_0, w_1), keys))$
		else
		  $epk =$ **OE.KeyRand**$(epk_{old}; w_1)$
		  store$(upk, (epk, gps_{old}, gss_{old}, E_{old}, b, witnesses :: (w_0, w_1), keys)$
		$sig \leftarrow$ **Sign**$(gsk, (epk, upk, epoch))$
		$cert = (upk, epk, epoch, sig)$

	$\xleftarrow{cert}$	

Fig. 2. The Subscribe $\leftrightarrow$ Enroll protocol

lawful access objection to strong cryptography and allowing its deployment, they make mass surveillance far more difficult.

Our approach has two major limitations. First, While accountable tracing signatures hold the authority accountable, they obviously only do so after the fact. An attacker who controls the authority gets access to all data until they are detected. Thus, if the goal is to maximize security, such systems should be avoided unless the alternatives are a non-accountable escrow system or no cryptographic protections at all. Second, the current scheme we have is by no means efficient and improving it either by using more efficient primitives or relaxing the security requirements is an area for future work.

References

[1] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 4–24, 2012.

[2] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 312–331, 2013.

[3] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 649–666, 2011.
[4] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010.
[5] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology—ASIACRYPT 2001*, pages 566–582. Springer, 2001.
[6] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology—Eurocrypt 2003*, pages 614–629. Springer, 2003.
[7] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479, pages 351–368, 2009.
[8] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265, 1991.
[9] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 226–247. Springer, 2009.
[10] Georg Fuchsbauer and Damien Vergnaud. Fair blind signatures without random oracles. In *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 16–33, 2010.
[11] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson. Universal re-encryption for mixnets. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, 2004.
[12] Matthew Green. Secure blind decryption. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2011.
[13] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In *CRYPTO*. Springer, 2012.
[14] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *Advances in Cryptology-EUROCRYPT 2004*, pages 571–589. Springer, 2004.
[15] Dennis Kügler and Holger Vogt. Auditable tracing with unconditional anonymity. 2001.
[16] Dennis Kügler and Holger Vogt. Offline payments with auditable tracing. In *Financial Cryptography*, pages 269–281. Springer, 2003.
[17] Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography*, pages 48–63. Springer, 2002.
[18] Jia Liu, Mark D Ryan, and Liqun Chen. Balancing societal security and individual privacy: Accountable escrow system. In *27th IEEE Computer Security Foundations Symposium (CSF)*, 2014.
[19] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society, 2013.

A Instantiation

We now detail how to instantiate our scheme using standard proof techniques.

A.1 Groth-Sahai proofs and simulation extraction

We will use Groth-Sahai (GS) proofs to efficiently instantiate the simulation-extractable proof system Π for the relation R_{sig} used in the signing algorithm of our accountable forward-tracing signature scheme. GS proofs operate over bilinear groups and we thus pick suitable primitives for SIG, OTS, and OE. In our instantiation, we let $\mathsf{GroupGen}$ on input security parameter 1^λ output the bilinear group $\mathcal{G} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, G, \hat{G})$ used by $\Pi.\mathsf{Setup}$. We let SIG, OTS, which will be pairing based, use the same groups, while OE will use $(\mathbb{G}_1, q, G)$ as its $(\mathbb{G}, q, g)$.

Recall the properties of bilinear groups:

- q is a λ-bit prime,
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are groups of prime order q with efficiently computable group operations, membership tests, and bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$,
- G and $\hat{G}$ are random generators of $\mathbb{G}_1$, $\mathbb{G}_2$, and $e(G, \hat{G})$ generates $\mathbb{G}_t$, and
- $\forall A \in \mathbb{G}_1, \forall B \in \mathbb{G}_2, \forall x, y \in \mathbb{Z} \ : \ e(A^x, B^y) = e(A, B)^{xy}$.
- an equation of the form $\prod_i \prod_j e(A_i, B_j)^{\gamma_{ij}} = 1$ for constants $\gamma_{ij} \in \mathbb{Z}_p$, and constants or variables $A_i \in \mathbb{G}_1$, $B_j \in \mathbb{G}_2$ is called a pairing product equation (PPE for short).

These are exactly the type of equations that can be proved using the GS proof system. Pairing product equations that are linear, that is equations in which variables appear only on one side of the pairing, are more efficient to prove.

We use the standard R-or-Sign technique to construct a simulation-extractable proof system Π from GS proofs, SIG and OTS. The left side of the 'or' proves the original statement (this is R_{sig} which we denote S_0), while the right side proves knowledge of a signature (σ_{sim}) under a public key in the *crs* (we denote this S_1). This signature certifies a one-time signature public key, which in turn is used to sign the instance and the Groth-Sahai proof. For details on the construction we refer to the construction of [7]. For the performance analysis we fall back on [2]:

The number of group elements in a proof of SE-NIZK is counted as follows. Let $S_0 = (R_{sig}(x,w) = 1)$ and $S_1 = (\mathsf{Verify}_{sim}(pk_{sim}, \sigma_{sim}, pk_{ots\_sim}) = 1)$ be the statements of the left and right side respectively, both represented by pairing product equations. The proof size of the SE-NIZK is as follows:

$$(\text{size of } S_0) + (\text{size of switch}) + (\text{size of } S_1) + (\text{size of } OTS)$$

$$= (\text{size of } S_0) \tag{1}$$

$$+ (|com| \times 1 + |\pi_{NL}| \times 1) \tag{2}$$

$$+ (|com| \times (|\sigma_{sim}| + S_1(C)) + |\pi_L| \times (S_1(L) + S_1(C)) + |\pi_{NL}| \times S_1(NL) \tag{3}$$

$$+ (|pk_{ots\_sim}| + |\sigma_{ots\_sim}|) \tag{4}$$

$|\pi_L|$ denotes the cost of proving a single linear PPE (e.g, $e(A,X) = 1$ where A is a constant),

$S_1(L)$ denotes the number of linear relations needed to prove $\mathsf{Verify}_{sim}(\ldots) = 1$

$|\pi_{NL}|$ denotes the cost of proving a non-linear PPE (e.g., $e(X,Y) = 1$)

$S_1(L)$ denotes the number of linear PPEs.

$S_1(C)$ denotes the number of constant pairings, e.g. $e(A,B)$, in PPEs.

pk_{ots_sim} is the size of the one-time signature key signed by the key in the *crs* (not part of a PPE)

σ_{ots_sim} is the size of the actual signature (again not part of a PPE).

For GS proofs over DLIN in asymmetric groups, we have $(|com|, |\pi_L|, |\pi_{NL}|) = (3,3,18)$; and for GS proofs over XDH, we have $(|com|, |\pi_L|, |\pi_{NL}|) = (2,2,8)$.

We consider the one-time simulation-extractable scheme SE-NIZK4 from [2] and an instantiation based on the xSIG scheme of [1] with unbounded simulation extraction. For these constructions, we summarize the overhead for achieving simulation extractability on top of simply proving the original statement, computed as the sum of equations (2)+(3)+(4), for both DLIN and XDH-based GS proofs in Table 1.

A.2 Putting it all together

We now examine the actual cost of our ATS scheme. It is dominated by proving the statement for the core relation R_{sig} of our accountable tracing signature scheme (S_0 above).

The equation for the proof size for R_{sig} is given below and the results for various instantiations is summarized in Table 2.

$$(\text{size of } S_0) \tag{5}$$

$$= |com| \times (|upk| + |epk| + |\sigma_{cert}| + |\sigma_u| + 1 + S_0(C)) \tag{6}$$

$$+ |\pi_L| \times (S_0(L) + S_0(C)) + |\pi_{NL}| \times S_0(NL) \tag{7}$$

Where $S_0(\{L, C, NL\})$ are defined as in the same way as $S_1(\{L, C, NL\})$

To reduce the size of *epk* we use a key-oblivious randomness-reusing variant of EIKO similar to [17] to allow us to efficiently encrypt multiple group elements. Effectively, each group element is encrypted under a distinct key using the same randomness. This results in a ciphertext overhead of only a single group element and thus shorter oblivious keys. (The PPE for verifiable encryption is $\bigwedge_{i=1}^{n} e(D_i, \hat{G}) = e(B_i, \hat{G}^s) e(m_i, \hat{G}) \wedge e(C, \hat{G}) = e(A, \hat{G}^s)$). Note that keys now correspond to encryptions of vectors of 1.

For performance, we instantiate SIG with two different schemes. As discussed above, we can for instance use a general purpose structure-preserving signature scheme (SPS) with secret key *csk* for the group manager and use an XRMA-secure scheme with secret keys *usk* for users.

As a further optimization, instead of the complete *upk*, we encrypt a single unique group element that serves as an identifier of *upk*. Instead of adding an additional group element to *cert*, the group manager can reuse one of the random elements of *upk* by ensuring that it is unique. The user encrypts this element together with σ_u, so OE needs to encrypt only $|\sigma_u| + 1$ group elements, thus $|epk| = |\sigma_u| + 2$ because of the ciphertext overhead.

An ATS signature σ consists of the elements $(\pi, \sigma_{ots}, pk_{ots}, escrw)$. To reduce the overhead, we generate only a single pk_{ots} (and one-time signature σ_{ots}) and use it both in S_0 and S_1. In both cases σ_{ots}, respectively σ_{ots_sim}, signed the instance, the proof, and the message. They can thus be merged.

The overall signature size of an ATS signature is thus the sum of the number of group elements needed to prove R_{sig}, the simulation-extraction overhead, and

SE-NIZK overhead	simulatability	$\lvert\sigma_{sim}\rvert$	$S_1(C)$	$S_1(L)$	$S_1(NL)$	$\lvert pk_{ots\_sim}\rvert$	$\lvert\sigma_{ots\_sim}\rvert$	DLIN	XDH
SE-NIZK4 [2]	one-time	3	2	3	0	2	2	55	34
SE-NIZK+xSIG [1]	unbounded	6	2	1	1	2×3	2	80	48

Table 1. Overhead of GS proof in terms of group elements.

Relation R_{sig}	$\lvert upk\rvert$	$\lvert epk\rvert$	$\lvert\sigma_{cert}\rvert$	$\lvert\sigma_u\rvert$	$S_0(C)$	$S_0(L)$	$S_0(NL)$	DLIN	XDH
SPS [3]+SPS [3]	1	$2+\lvert\sigma_u\rvert$	3	3	$8+\lvert\sigma_u\rvert$	$4+\lvert\sigma_u\rvert$	4	198	116
SPS [3]+xSIG [1]	10	$2+\lvert\sigma_u\rvert$	3	6	$6+\lvert\sigma_u\rvert$	$3+\lvert\sigma_u\rvert$	3	237	144
SIG2 [1]+xSIG [1]	10	$2+\lvert\sigma_u\rvert$	14	6	$4+\lvert\sigma_u\rvert$	$4+\lvert\sigma_u\rvert$	4	279	170

Table 2. Number of group elements needed for ATS relation R_{sig}.

the ciphertext size $|escrw| = |epk|$. In our example instantiations it ranges from 155 to 367 group elements.

Because of the availability of a large number of structure-preserving primitives, there is plenty of room for optimization, especially when one is aiming for both efficiency and weak cryptographic assumptions. We stress, moreover, that our instantiation does not make use of random oracles in its security proof.

B Proofs

B.1 Security for ATS.

Theorem 13 (Anonymity under tracing)**.** *If* Π *is zero-knowledge and simulation-extractable and* OE *is plaintext indistinguishable and key private, then the construction described in Section 4 is anonymous under tracing as defined in Section 3.2.*

Proof. We now proceed to describe a sequence of hybrid experiments.

- $\eth_0$. The original AuT game.
- $\eth_1$. Same as $\eth_0$, except that in the Ch oracle we use Sim to simulate π_b in σ_b and store the simulated proof along with its inputs in a log LS. By the zero-knowledge property of the proof system, the attacker has a negligible advantage in distinguishing between this and the previous game.
- $\eth_2$. Same as $\eth_1$, except that for proofs $\pi \in LS$ we answer with the stored data from the simulator. For $\pi \notin LS$ we use extraction to answer OPEN oracle queries without using the decryption key in gsk. If this fails we abort. Because Π is simulation extractable, by definition the probability of failing to extract on a proof that was not directly produced by the simulator (i.e. is not in LS) is negligible, and hence so is the probability of abort and the attacker's advantage in distinguishing between this and the previous game.
- $\eth_3$ and $\eth_4$ correspond to $\eth_2$ and $\eth_3$ of the anonymity with accountability proof.

In $\eth_4$, all inputs to $\mathcal{A}$ are independent of b and its advantage is therefore zero. □

Theorem 14 (Traceability)**.** *If* Π *is extractable,* SIG *is unforgeable, and* OTS *is strongly unforgeable, then the construction described in Section 4 is traceable as defined in Section 3.2.*

Proof. We proceed through a sequence of hybrids.

- $\eth_0$. The original traceability game.
- $\eth_1$. The same as $\eth_0$, except that we attempt to extract $(upk, epk, \sigma_{cert}, \sigma_u)$ from the proof π in σ and abort without $\mathcal{A}$ winning if extraction fails. The difference in the advantage of $\mathcal{A}$ winning compared with $\eth_0$ is negligible by the extractability of the proof system.
- $\eth_2$. Same as $\eth_1$, except that we abort if the attacker uses a signature σ_{cert} on a fresh upk, epk.

  Lemma 15. *The difference in the advantage of* $\mathcal{A}$ *winning compared with* $\eth_1$ *is negligible by the unforgeability of* SIG.

- $\eth_3$. Same as $\eth_2$, except that we abort if the attacker uses a signature σ_u on a fresh OTS public key pk_{ots}, i.e., one that did not result from a signing query.

  Lemma 16. *The difference in the advantage of* $\mathcal{A}$ *winning compared with* $\eth_2$ *is negligible by the unforgeability of* SIG.

- $\Game_4$. Same as $\Game_3$, except that we abort if the attacker reuses a OTS public key to sign a different message (i.e not the one stored in Q). The probability of this happening, and hence aborting, is negligible by a slight variant of Lemma 15 and 16 for one-time signatures.

In $\Game_4$ the advantage of $\mathcal{A}$ is zero. □

Theorem 17 (Non-frameability). *Let* SIG *be an unforgeable signature scheme and* OTS *a strongly unforgeable one-time signature scheme, then the construction described in Section 4 is non-frameable as defined in Section 3.2.*

Proof. Informally, causing Judge to validate is the same as forging one of the two signature in ψ. This is assumed impossible for secure signature schemes.

The proof is thus nearly identical to that of traceability, except that $\mathcal{A}$ already provides σ_u as part of his output and it is thus does not need to extracted. For the hybrids see $\Game_3$ and $\Game_4$ of the traceability proof. □

Theorem 18 (Anonymity with accountability). *If* $\mathsf{\Pi}$ *is zero-knowledge and* OE *is plaintext indistinguishable and key private, then the construction described in Section 4 is accountably anonymous as defined in Section 3.2.*

Proof. We proceed through a sequence of hybrids.

- $\Game_0$. The original AwA game.
- $\Game_1$. Same as $\Game_0$, except that we replace the zero-knowledge proofs by simulated proofs. By the zero-knowledge property of the proof system, the attacker has a negligible advantage in distinguishing between this and the previous game.
- $\Game_2$. Same as $\Game_1$, except we modify $\mathsf{ATS.Sign}$ in game Ch to produce an escrow ciphertext of the encryption of 0 for σ_b.

  Lemma 19. *By the INDr-CPA property of* EIKO*, the new ciphertext is indistinguishable from the old one and so* $\Game_1$ *and* $\Game_2$ *are indistinguishable.*

- $\Game_3$. Same as $\Game_2$, except we modify $\mathsf{ATS.Sign}$ in game Ch to use a fresh random key as the escrow key.

  Lemma 20. *By the key privacy property of* EIKO*, the new ciphertext is indistinguishable from the old one and so* $\Game_3$ *and* $\Game_4$ *are indistinguishable.*

In $\Game_3$, Ch returns a simulated proof and an encryption of zero under a new key. Hence all inputs to $\mathcal{A}$ are independent of b and its advantage is therefore zero. □

Theorem 21 (Trace-obliviousness). *If* OE *is key randomizable, and* $\mathsf{\Pi}$ *is extractable, then the construction described in Section 4 is trace-oblivious as defined in Section 3.2.*

Proof. We proceed through a sequence of hybrids.

- $\Game_0$. The original trace-obliviousness game.
- $\Game_1$. Same as $\Game_0$, except that we change OPEN to invoke the extractor for $\mathsf{\Pi}$ to decrypt $escrw$ instead of using the private key. The attacker has a negligible advantage in distinguishing between this game and $\Game_0$ by the extractability of the proof system.
- $\Game_2$. Same as $\Game_1$, except that we change Enroll in ENROLL to return freshly generated public keys instead of randomized keys.

  Lemma 22. *By the key randomizability property of* EIKO*, the new key is indistinguishable from the old one.*

In $\Game_2$ the output of all oracles is independent of b, and therefor the adversary's advantage is zero. □

B.2 Proofs of supporting lemmas

Proof Lemma 4. Given an attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ who breaks KPr we construct a distinguisher $\mathcal{D}$ for DDH.

$$
\begin{array}{l}
\textbf{Distinguisher } \mathcal{D}(\mathbb{G}, X, Y, T) \triangleq \\
b \leftarrow \{0,1\} \\
\lambda, \mu, \xi, r_0, r_1, \leftarrow \mathbb{Z}_q \\
X_0 \leftarrow X; Y_0 \leftarrow Y; T_0 \leftarrow T \\
X_1 \leftarrow X \cdot g^{\lambda}; Y_1 \leftarrow Y^{\xi} \cdot g^{\mu}; T_1 \leftarrow T^{\xi} \cdot X^{\mu} \cdot Y^{\lambda\xi} \cdot g^{\lambda\xi} \\
pk_0 \leftarrow (g^{r_0}, Y_0^{r_0}) \\
pk_0 \leftarrow (g^{r_1}, Y_1^{r_1}) \\
(m, pk_0', w_0, pk_1', w_1, st) \leftarrow \mathcal{A}_0(\mathcal{G}, pk_0, pk_1) \\
\textbf{if } pk_0' \neq \mathsf{KeyRand}(pk_0; w_0) \vee pk_1' \neq \mathsf{KeyRand}(pk_1; w_1) \\
\quad \textbf{return } \perp \\
d \leftarrow \mathcal{A}_1(X^{r_b w_b}, T_b^{r_b w_b} \cdot M, st) \\
\textbf{return } (d = b)
\end{array}
$$

Given Diffie-Hellman's random self-reducibility, if X, Y, T is a Diffie-Hellman triple, then so is X_1, Y_1, T_1. Moreover, both triples are distributed identically regardless and produce the proper distributions of keys in EIKO. In the case that T is a random group element, then the challenge ciphertext given to A contains no information, as Y_b and $T_b^{w_b}$ are identically distributed regardless of b. Hence, $\mathcal{A}$'s advantage at guessing the bit is negligible. On the other hand, in the case where the challenge is a valid Diffie-Hellman triple, $\mathcal{A}$'s inputs are the same as in **Game** KO, since $T_b^{r_b w_b} = (g^{x_b \cdot y_b})^{r_b \cdot w_b} = g^{x_b \cdot y_b \cdot r_b \cdot w_b} = (X^{r_b \cdot w_b})^{y_b} = C^{y_b}$.

Thus if $\mathcal{A}$ has a non-negligible advantage at breaking **Game** KO, then $\mathcal{D}$ breaks DDH. □

Proof Lemma 5. Given a DDH challenge $(\mathbb{G}, X, Y, T)$, the reduction is immediate.

$$
\begin{aligned}
&\textbf{Distinguisher } \mathcal{D}(\mathbb{G}, X, Y, T) \triangleq \\
&r, \leftarrow \mathbb{Z}_q \\
&d \leftarrow \mathcal{A}((g^r, X^r), (Y^r, T^r)) \\
&\textbf{return } d
\end{aligned}
$$

In the case where $(X = g^x, Y = g^y, T = g^{xy})$, then our original key is (g^r, g^{xr}), and the second one is $((g^r)^y, (g^{xr})^y))$, i.e., the original key re-randomized by y. On the other hand, where $(X = g^x, Y = g^y, T = g^z)$, the second key is unrelated. Thus if $\mathcal{A}$ distinguishes between a real or random key with a non-negligible advantage, then the distinguisher above breaks DDH with the same advantage. □

Proof Lemma 6. Given an attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ who wins against INDr we construct a DDH distinguisher $\mathcal{D}$.

$$
\begin{aligned}
&\textbf{Distinguisher } \mathcal{D}(\mathbb{G}, X, Y, T) \triangleq \\
&b \leftarrow \{0, 1\} \\
&r \leftarrow \mathbb{Z}_q \\
&pk \leftarrow (g^r, X^r) \\
&pk', r', m_0, m_1 \leftarrow \mathcal{A}_0(\mathcal{G}, pk) \\
&\textbf{if } pk' \neq \mathsf{KeyRand}(pk; r') \textbf{ then} \\
&\quad \textbf{return } \bot \\
&ct \leftarrow ((Y^r)^{r'}, (T^r)^{r'} \cdot m_b) \\
&b' \leftarrow \mathcal{A}_1(ct) \\
&\textbf{return } (b = b')
\end{aligned}
$$

In the case where (X, Y, T) is a DDH triple, $\mathcal{A}$'s view is identical to INDr game: she receives a public key g^{rx} and a base g^r and then a properly formed ciphertext $\left(((g^r)^{r'})^y, ((g^r)^{r'})^{xy} \cdot m\right)$. On the other hand, when T is a random group element, $(T^r)^{r'} \cdot m_b$ reveals no information about m_b. Hence her advantage is negligible. Thus if A has an advantage in winning INDr, then the above breaks DDH with the same advantage. □

Proof Lemma 15. Given an attacker $\mathcal{A}_\Game$ who forges a signature in Trace, we construct a reduction to the standard EU-CMA signature game where an attacker A is given access to a signing oracle and a public key and produces a forgery on a message not previously signed by the oracle.

$\mathcal{A}^{\mathsf{SIG}}(pk)$ works as follows. Given pk it simulates the standard Trace game using the verification key it received as input as the authority's key in the parameters. We modify SIGN to return signatures generated by the oracle SIG oracle. When $\mathcal{A}_\Game$ triggers abort by producing a signature on a key not in S, $\mathcal{A}$ returns it as a signature forgery. □

Proof Lemma 16. This proof is nearly identical to that of Lemma 15. Instead of inserting the challenge public key into the parameters, however, we must have ENROLL embed this key in some generated user key. Unfortunately, we can only do so for one query and must simply blindly guess upon which one to do so. Having done so, the game continues as in the proof above and, if we guessed correctly, we get the appropriate forgery. Because $\mathcal{A}_\Game$ makes at most $poly()$ queries to ENROLL, there is a $\frac{1}{poly()}$ chance $\mathcal{A}_\Game$ forges a signature under the target key (i.e., that the forgery resulted from the query we picked), thus $\mathcal{A}_\Game$'s probability of abort is negligible. □

Proof Lemma 19. $\mathcal{A}_\Game$ makes at most $poly()$ queries to the Ch oracle. We construct a series of hybrids—$\Game_2^0, \ldots, \Game_2^i, \ldots, \Game_2^{poly()}$ where $\Game_2^{poly()}$ is equivalent to $\Game_2$—replacing the ciphertext in each successive query with 0. Given an adversary $\mathcal{A}_\Game^i$ who detects the hybrid that modifies the ith query to Ch, we construct an adversary $\mathcal{A}_{IND} = (\mathcal{A}_0, \mathcal{A}_1)$ who breaks INDr as follows.

$\mathcal{A}_0$ runs the standard AwA game with $\mathcal{A}_\Game^i$ using the provided parameters as the (non)escrow key in gp. On the ith query to the Ch oracle, $\mathcal{A}_0$ returns $(cert_b.epk, w_b^{escrw}, ctm, 0)$ as (pk', r, m_0, m_1) where ctm is the correct content of an escrow ciphertext. Upon receiving the challenge ciphertext, $\mathcal{A}_1$ constructs σ using ct as the escrow ciphertext and allows the AwA game to continue. Finally, it returns the resulting bit.

In the case where m_0 is chosen in the $INDr$ game, $\mathcal{A}_\Game^i$'s view is identical to that of $\Game_2^{i-1}$ (i.e. where the ith query is untampered with and results in a proper ciphertext). On the other hand, in the case where m_1 is chosen, her view is identical to the case of $\Game_2^i$ (i.e., where the ith ciphertext is an encryption of the all zero string). Thus $\mathcal{A}_\Game^i$'s advantage is the same as $\mathbf{Adv}^{\mathsf{INDr}}(\mathcal{A})$ which is negligible. Thus $\mathcal{A}_\Game$'s advantage for the whole set of substitutions is $\mathbf{Adv}^{\mathsf{INDr}}(\mathcal{A}) \cdot poly()$ which is still negligible. □

Proof Lemma 20. This proof proceeds similarly to Lemma 19. Again, $\mathcal{A}_{\Game}$, this time distinguishing between $\Game_2$ and $\Game_3$, makes at most polynomially many queries to Ch. We construct a series of hybrids —$\Game_3^0, \ldots \Game_3^i$, $\ldots \Game_3^{poly()}$ where $\Game_3^{poly()}$ is equivalent to $\Game_3$—in which we swap the key in the ith query. Given an adversary $\mathcal{A}_{\Game}^i$ who detects the hybrid that modifies the ith query to Ch, we construct an adversary $\mathcal{A}_{KP} = (\mathcal{A}_0, \mathcal{A}_1)$ who breaks KPr as follows.

$\mathcal{A}_0(pk_0, pk_1)$ runs the standard AwA game with $\mathcal{A}_{\Game}$ using pk_0 as the key in gp. On the ith query to the Ch oracle, $\mathcal{A}_0$ samples fresh randomness r and returns $(0, cert_b.epk, w_0^{escrw}, \mathsf{KeyRand}(pk_1; r), r, 0)$ as $(m, pk_0', w_0, pk_1', w_1, st)$. Upon receiving the challenge ciphertext, $\mathcal{A}_1$ constructs σ using ct as the escrow ciphertext and allows the AwA game to continue. Finally, it returns the resulting bit.

In the case where pk_0 is chosen as the encryption key in the KPr game, $A_{\Game}^i$'s view is identical to $\Game_3^{i-1}$, where the provided key is used. On the other hand, in the case where pk_2 is chosen, her view is identical to $\Game_3^i$ where a fresh key is used. Thus her advantage in distinguishing between any two hybrids is $\mathbf{Adv}^{\mathsf{KPr}}(\mathcal{A})$ which is negligible. This implies $\mathcal{A}_{\Game}$'s advantage $\mathbf{Adv}^{\mathsf{KPr}}(\mathcal{A}) \cdot poly()$ which is still negligible. □

Proof Lemma 22. This proof proceeds similarly to the others above. We construct a series of hybrids, one for each query to ENROLL where we sequentially replace the returned key with a random one. We denote these $\Game_2^0, \ldots \Game_2^i, \ldots \Game_2^{poly()}$ where $\Game_2^{poly()}$ is equivalent to $\Game_2$. Given an adversary $\mathcal{A}_{\Game}^i$ who detects the hybrid that tampers with the ith query, We construct $\mathcal{A}_{KR}(pk, pk_b)$ as follows.

$\mathcal{A}_{KR}$ embeds pk as the escrow key in the parameters and runs $A_{\Game}^i$. On the query to ENROLL, it returns pk_b. The game continues as normal. Finally, it returns the resulting bit.

In the case of $\mathcal{A}_{KR}(pk, pk_0)$, $A_{\Game}^i$ receives a randomized key and her view is identical to that of $\Game_2^{i-1}$. On the other hand, in the case of $\mathcal{A}_{KR}(pk, pk_1)$, her view is identical to that in $\Game_2^i$. Thus her advantage is $\mathbf{Adv}^{\mathsf{KR}}(\mathcal{A})$ when detecting tampering with any one query. $\mathcal{A}_{\Game}$'s advantage in distinguishing between $\Game_1$ and $\Game_2$ is $\mathbf{Adv}^{\mathsf{KR}}(\mathcal{A}) \cdot poly()$ which is negligible. □