Abuse Resistant Law Enforcement Access Systems

Matthew Green[1(✉)], Gabriel Kaptchuk[2], and Gijs Van Laer[1]

[1] Johns Hopkins University, Baltimore, USA
mgreen@cs.jhu.edu, gijs.vanlaer@jhu.edu
[2] Boston University, Boston, USA
kaptchuk@bu.edu

Abstract. The increasing deployment of end-to-end encrypted communications services has ignited a debate between technology firms and law enforcement agencies over the need for lawful access to encrypted communications. Unfortunately, existing solutions to this problem suffer from serious technical risks, such as the possibility of operator abuse and theft of escrow key material. In this work we investigate the problem of constructing law enforcement access systems that mitigate the possibility of unauthorized surveillance. We first define a set of desirable properties for an abuse-resistant law enforcement access system (ARLEAS), and motivate each of these properties. We then formalize these definitions in the Universal Composability (UC) framework, and present two main constructions that realize this definition. The first construction enables *prospective* access, allowing surveillance only if encryption occurs after a warrant has been issued and activated. The second, more powerful construction, allows *retrospective* access to communications that occurred prior to a warrant's issuance. To illustrate the technical challenge of constructing the latter type of protocol, we conclude by investigating the minimal assumptions required to realize these systems.

1 Introduction

Communication systems are increasingly deploying end-to-end (E2E) encryption as a means to secure physical device storage and communications traffic. E2E encryption systems differ from traditional link encryption mechanisms in that keys are not available to service providers, but are instead held by endpoints: typically end-user devices such as phones or computers. This approach ensures that plaintext data cannot be accessed by providers and manufacturers, or by attackers who may compromise their systems. Widely-deployed examples include messaging protocols [6,73,78], telephony [4], and device encryption [5,43], with some systems deployed to billions of users.

The adoption of E2E encryption in commercial services has provoked a backlash from the law enforcement and national security communities around the world, based on concerns that encryption will hamper agencies' investigative

A. Canteaut and F.-X. Standaert (Eds.): EUROCRYPT 2021, LNCS 12698, pp. 553–583, 2021.
https://doi.org/10.1007/978-3-030-77883-5_19

and surveillance capabilities [10,36,77]. The U.S. Federal Bureau of Investigation has mounted a high-profile policy campaign called "Going Dark" around these issues [34], and similar public outreach has been conducted by agencies in other countries [55]. These campaigns have resulted in legislative proposals in the United States [46,66,71] that seek to discourage the deployment of "warrant-proof" end-to-end encryption, as well as adopted legislation in Australia that requires providers to guarantee access to plaintext in commercial communication systems [76].

The various legislative proposals surrounding encryption have ignited a debate between technologists and policymakers. Technical experts have expressed concerns that these proposals, if implemented, will undermine the security offered by encryption systems [1,61,74], either by requiring unsafe changes or prohibiting the use of E2E encryption altogether. Law enforcement officials have, in turn, exhorted researchers to develop new solutions that resolve these challenges [10]. However, even the basic technical requirements of such a system remain unspecified, complicating both the technical and policy debates.

Existing Proposals for Law Enforcement Access. A number of recent and historical technical proposals have been advanced to resolve the technical questions raised by the encryption policy debate [13,14,30,55,68,75,79]. With some exceptions, the bulk of these proposals are variations on the classical *key escrow* [31] paradigm. In key escrow systems, one or more trusted authorities retain key material that can be used to decrypt targeted communications or devices.

Technologists and policymakers have criticized key escrow systems [1,33,62], citing concerns that, without additional protection measures, these systems could be abused to covertly conduct mass surveillance of citizens. Such abuses could result from a misbehaving operator or a compromised escrow keystore. Two recent policy working group reports [33,62] provide evidence that, at least for the case of communications services, these concerns are shared by members of the policy and national security communities.[1] Reflecting this consensus, recent high-profile technical proposals have limited their consideration only to the special case of *device encryption*, where physical countermeasures (*e.g.,* physical possession of a device, tamper-resistant hardware) can mitigate the risk of mass surveillance [14,68]. Unfortunately, expanding the same countermeasures to messaging or telephony software seems challenging.

Abuse of Surveillance Mechanisms. Escrow-based access proposals suffer from three primary security limitations. First, key escrow systems require the storage of valuable key material that can decrypt most communications in the system. This material must be accessible to satisfy law enforcement request, but must simultaneously be defended against sophisticated, nation-state supported

[1] The Carnegie Institution report [33] concludes that "In the case of data in motion, for example, our group could identify no approach to increasing law enforcement access that seemed reasonably promising to adequately balance all of the various concerns".

attackers. Second, in the event that key material is surreptitiously exfiltrated from a keystore, it may be difficult or impossible to detect its subsequent misuse. This is because escrow systems designed to allow lawful access to encrypted data typically store *decryption keys*, which can be misused without producing any detectable artifact.[2] Finally, these access systems require a human operator to interface between the digital escrow technology and the non-digital legal system, which raises the possibility of misbehavior by operators. These limitations must be addressed before any law enforcement access system can be realistically considered, as they are not merely theoretical: wiretapping and surveillance systems have proven to be targets for both nation-state attacks and operator abuse [19,44,60].

Overcoming these challenges is further complicated by law enforcement's desire to access data that was encrypted before an investigation is initiated. For example, several recent investigations requested the unlocking of suspects' phones or message traffic in the wake of a crime or terrorist attack [56]. Satisfying these requests would require *retrospectively* changing the nature of the encryption scheme used: ciphertext must be strongly protected before an investigation begins, but they must become accessible to law enforcement after an investigation begins. Satisfying these contradictory requirements is extraordinarily challenging without storing key material that can access all past ciphertexts, since a ciphertext may be created *before* it is known if there will be a relevant investigation in the future.

Law enforcement access systems that do not fail open in the face of lost key material or malicious operators have been considered in the past, *e.g.,* [13,16,79]. Bellare and Rivest [13] proposed a mechanism to build *probabilistic* law enforcement access, in order to mitigate the risk of mass surveillance. Wright and Varia [79] proposed cryptographic puzzles as a means to increase the financial cost of abuse. While these might be theoretically elegant solutions, such techniques have practical limitations that may hinder their adoption: law enforcement is unlikely to tolerate arbitrary barriers or prohibitive costs that might impede legitimate investigations. Moreover, these proposals do little to enable detection of key theft or to prevent more subtle forms of misuse.

Towards Abuse Resistant Law Enforcement Access. In this work, we explore if it is technically possible to limit abuse while giving law enforcement the capabilities they are truly seeking: quickly decrypting relevant ciphertexts during legally compliant investigations. To do this, we provide a new cryptographic definition for an *abuse resistant law enforcement access system*. This definition focuses on abuse resistance by weaving *accountability* features throughout the access process. More concretely, our goal is to construct systems that realize the following three main features:

[2] This contrasts with the theft of *e.g.,* digital certificates or signing keys, where abuse may produce artifacts such as fraudulent certificates [64] or malware artifacts that can be detected through Internet-wide surveillance.

- **Global Surveillance Policies.** To prohibit abuse by authorized parties, access systems must enforce specific and *fine-grained* global policies that restrict the types of surveillance that may take place. These policies could, for example, encompass limitations on the number of messages decrypted, the total number of targets, and the types of data accessed. They can be agreed upon in advance and made publicly available. This approach ensures that global limits can be developed that meet law enforcement needs, while also protecting the population against unlimited surveillance.
- **Detection of Abuse.** We require that any unauthorized use of escrow key material can be detected, either by the public or by authorized auditing parties. Achieving this goal ensures that even fully-adversarial use of escrow key material (*e.g.,* following an undetected key exfiltration) can be detected, and the system's security can be renewed through rekeying.
- **Operability.** At the same time, escrow systems must remain *operable*, in the sense that honest law enforcement parties should be able to access messages sent through a compliant system. We aim to guarantee this feature by ensuring that it is easy to verify that a message has been correctly prepared.

We stress that the notion of abuse-resistance is different from *impossible to abuse*. Under our definitions abuse may still happen, but the features described above will allow the abuse to be quickly identified and system security renewed. The most critical aspect of our work is that we seek to enforce these features *through the use of cryptography*, rather than relying on correct implementation of key escrow hardware or software, or proper behavior by authorities.

Prospective vs. Retrospective Surveillance. We will divide the access systems we discuss into two separate categories: *prospective* and *retrospective*. When using a *prospective* system, law enforcement may only access information encrypted sent or received from suspects *after* those suspects have been explicitly selected as targets for surveillance: this is analogous to "placing an alligator clip on a wire" in an analog wiretap. A *retrospective* access system, as described above, allows investigators to decrypt past communications, even those from suspects who were not the target of surveillance when encryption took place. Retrospective access clearly offers legitimate investigators more capabilities, but may also present a greater risk of abuse. Indeed, achieving accountable access in the challenging setting of retrospective key escrow, where encryption may take place *prior* to any use of escrow decryption keys, is one of the most technically challenging aspects of this work.

Our Contributions. More concretely, in this work we make the following contributions.

- **Formalizing security notions for abuse resistant law enforcement access systems.** We first provide a high-level discussion of the properties required to prevent abuse in a key escrow system, with a primary focus on the general data-in-motion setting: *i.e.,* we do not assume that targets possess trusted hardware. Based on this discussion, we formalize the roles and

protocol interface of an Abuse-Resistant Law Enforcement Access System (ARLEAS): a message transmission framework that possesses law enforcement access capability with strong accountability guarantees. Finally, we provide an ideal functionality $\mathcal{F}_{\text{ARLEAS}}$ in Canetti's Universal Composability framework [21].

- **A prospective ARLEAS construction from non-interactive secure computation.** We show how to realize ARLEAS that is restricted to the case of *prospective* access: this restricts the use of ARLEAS such that law enforcement must commit to surveillance parameters before a target communication occurs. Each message contains a message for a non-interactive secure computation protocol [49] that will release plaintext only if law enforcement has activated a relevant warrant before encryption. We note that more simple and efficient constructions are possible if restrictions are put on warrants, *e.g.* warrants must list specific receivers; due to space constraints, we discuss these approaches in the full version of the paper.
- **A retrospective ARLEAS construction from proof-of-publication ledgers and extractable witness encryption.** We show how to realize ARLEAS that admits *retrospective* access, while still maintaining the auditability and detectability requirements of the system. The novel idea behind our construction is to use secure *proof-of-publication ledgers* to condition cryptographic escrow operations. The cryptographic applications of proof-of-publication ledgers have recently been explored (under slightly different names) in several works [25,45,51,69]. Such ledgers may be realized using recent advances in consensus networking, a subject that is part of a significant amount of research.
- **Evaluating the difficulty of retrospective systems.** Finally, we investigate the *minimal* assumptions for realizing retrospective access in an accountable law enforcement access system. As a concrete result, we present a lower-bound proof that any protocol realizing retrospective ARLEAS implies the existence of an extractable witness encryption scheme for some language L which is related to the ledger functionality and policy functions of the system. While this proof does not imply that all retrospective ARLEAS realizations require extractable witness encryption for general languages (*i.e.,* it may be possible to construct languages that have trivial EWE realizations), it serves as a guidepost to illustrate the barriers that researchers may face in seeking to build accountable law enforcement access systems.

1.1 Towards Abuse Resistance

In this work we consider the problem of constructing secure message transmission protocols with abuse resistant law enforcement access, which can be seen as an extension of secure message transmission as formalized in the UC framework by Canetti [21,22]. Before discussing our technical contributions, we present the parties that interact with such a system and discuss several of the security properties we require.

The ARLEAS Setting. An ARLEAS system is comprised of three types of parties:

1. **Users:** Users employ a secure message transmission protocol to exchange messages with other users. From the perspective of these users, this system acts like a normal messaging service, with the additional ability to view public audit log information about the use of warrants on information sent through the system.
2. **Law Enforcement:** Law enforcement parties are responsible for initiating surveillance and accessing encrypted messages. This involves determining the scope of a surveillance request, obtaining a digital warrant, publishing transparency information, and then accessing the resulting data.
3. **Judiciary:** The final class of parties act as a check on law enforcement, determining whether a surveillance request meets the necessary legal requirements. In our system, any surveillance request must be approved by a judge before it is activated on the system. In our model we assume a single judge per system, though in practice this functionality can be distributed.

At setup time an ARLEAS system is parameterized by three functions, which we refer to as the global policy function, $p(\cdot)$, the warrant transparency function, $t(\cdot)$, and the warrant scope check function, $\theta(\cdot)$.[3] The purpose of these functions will become clear as we discuss operation and desired properties below. Finally, our proposals assume the existence of a verifiable, public broadcast channel, such as an append-only ledger. While this ledger may be operated by a centralized party, in practice we expect that such systems would be highly-distributed, *e.g.* using blockchain or consensus network techniques.

ARLEAS Operation. To initiate a surveillance request, law enforcement must first identify a specific class of messages (*e.g.* by metadata or sender/receiver); it then requests a surveillance warrant w from a judge. The judge reviews the request and authorizes or rejects the request. If the judge produces an authorized warrant, law enforcement must take a final step to *activate* the warrant in order to initiate surveillance. This activation process is a novel element of an abuse resistant access scheme, and it is what allows for the detection of misbehavior. To enforce this, we require that activation of a warrant w results in the publication of some information that is viewable by all parties in the system. This information consists of two parts: (1) a proof that the warrant is *permissible* in accordance with the global policy function, *i.e.* $p(w) = 1$, and (2) some transparency data associated with the warrant. The amount and nature of the transparency data to be published is determined by the warrant transparency function $t(w)$. Once the warrant has been activated, and the relevant information has been made public, law enforcement will be able to access any message that is within the scope of the warrant, as defined by the warrant scope check function $\theta(w)$.

[3] We later introduce a fourth parameterizing function, but omit it here for the clarity of exposition.

1.2 Technical Overview

We now present an overview of the key technical contributions of this work. We will consider this in the context of secure message transmission systems, which can be generalized to the setting of encrypted storage. Our overview will begin with intuition for building prospective ARLEAS, and then we will proceed to retrospective ARLEAS.

Accountability From Ledgers. For an ARLEAS the most difficult properties to satisfy are accountability and detectability. Existing solutions attempt to achieve this property by combining auditors and key escrow custodians; in order to retrieve key material that facilitates decryption, law enforcement must engage with an auditor. This solution, however, does not account for dishonest authorities, and is therefore vulnerable to covert key exfiltration and collusion. In our construction, we turn to public ledgers—a primitive that can be realized using highly-decentralized and auditable systems—as a way to reduce these trust assumptions.

Ledgers have the property that any party can access their content. Importantly, they also have the property that any parties can be convinced that other parties have access to these contents. Thus, if auditing information is posted on a ledger, all parties are convinced that that information is truly public. We note that using ledgers in this way is fundamentally different from prior work addressing encrypted communications; our ledger is a public functionality that does not need to have any escrow secrets. As such, if it is corrupted, there is no private state that can be exploited by an attacker.

Warm up: Prospective ARLEAS. To build to our main construction, we first consider the simpler problem of constructing a *prospective* access system, one that is capable of accessing messages that are sent subsequent to a warrant being activated.

A key aspect of this construction is that we consider a relatively flexible setting where parties have network access, and can receive periodic communications from escrow system operators prior to transmitting messages. We employ a public ledger for transmission of these messages, which provides an immutable record as well as a consistent view of these communications. The goal in our approach is to ensure that escrow updates embed information about the specifics of surveillance warrants that are active, while ensuring that even corrupted escrow parties cannot abuse the system.

Prospective ARLEAS for Arbitrary Predicates. The core intuition of our approach is to construct a "dual-trapdoor" public-key encryption system that senders can use to encrypt messages to specific parties. This scheme is designed with two ciphertexts c_1 and c_2, such that c_1 can be decrypted by the intended recipient using a normal secret key, while c_2 can be decrypted by law enforcement only if the recipient is under active surveillance, *i.e.* law enforcement has a warrant w that applies to the message and has posted any necessary transparency information. A feature of this scheme is that for all recipients not the target of surveillance, c_2 should leak no information about the plaintext to

law enforcement. In this work, we use non-interactive secure computation (NISC) [49], a reusable, non-interactive version of two-party computation to "encrypt" the ciphertext c_2. NISC for an arbitrary function f allows a receiver to post an encryption of some secret x_1 such that all players can reveal $f(x_1, x_2)$ to the receiver with only one message, without revealing anything about x_2 beyond the output of the function.

In prospective surveillance, law enforcement must activate their warrant before it can be used to decrypt traffic. When activating a warrant, law enforcement computes the transparency information for their warrant $\mathsf{info} \leftarrow t(w)$ along with the first message of the NISC scheme, embedding the warrant, and posts these onto the ledger. Whenever a sender sends a message m, they retrieve law enforcement's latest post, generate c_1 as using a normal public key encryption scheme and then generate c_2 which, using the NISC scheme, allows law enforcement to compute $f(w, (m, \mathsf{meta})) = m \wedge \theta(\mathsf{meta}, w)$, where $\theta(\cdot, \cdot)$ evaluates if the warrant applies to this particular message (we will discuss $\theta(\cdot, \cdot)$ in more detail in Sect. 3). Notice that if $\theta(\mathsf{meta}, w) = 0$, then the output of the NISC evaluation is uncorrelated with the message. However, if $\theta(\mathsf{meta}, w) = 1$, meaning law enforcement has been issued a valid warrant, then the message is recovered. We note that it is possible to construct a more concretely efficient scheme that uses lossy encryption instead of NISC, as long as warrants specify the *identity* of users; we discuss this construction in the full version of the paper.

From Prospective to Retrospective. The major limitation of the ARLEAS construction above is that it is fundamentally restricted to the case of *prospective* access. Abuse resistance derives from the fact that "activation" of a warrant results in a distribution of fresh encryption parameters to users, and each of these updates renders only a subset of communications accessible to law enforcement. A second drawback of the prospective protocol is that it requires routine communication between escrow authorities and the users of the system, which may not be possible in all settings.

Updating these ideas to provide *retrospective* access provides a stark illustration of the challenges that occur in this setting. In the retrospective setting, the space of targeted communications is unrestricted at the time that encryption takes place. By the time this information is known, both sender and recipient may have completed their interaction and gone offline. Using some traditional, key based solution to this problem implies the existence of powerful master decryption keys that can access *every* ciphertext sent by users of the system. Unfortunately, granting such power to any party (or set of parties) in our system is untenable; if this key material is compromised, any message can be decrypted without leaving a detectable artifact. The technical challenge in the retrospective setting is to find an alternative means to enable decryption, such that decryption is only possible on the conditions that (1) a relevant warrant has been issued that is compliant with the global policy function, (2) a detectable artifact has been made public. This mechanism must remain secure even when encryption occurs significantly before the warrant is contemplated.

Ledgers as a Cryptographic Primitive. A number of recent works [24,25,45,51,69] have proposed to use public ledgers as a means to *condition* cryptographic operations on published events. This paradigm was initially used by Choudhuri *et al.* [25] to achieve fairness in MPC computations, while independently a variant was proposed by Goyal and Goyal [45] to construct one-time programs without the need for trusted hardware. Conceptually, these functionalities all allow decryption or program execution to occur only *after* certain information has been made public. This model assumes the existence of a secure global ledger $\mathcal{L}$ that is capable of producing a publicly-verifiable proof π that a value has been made public on the ledger. In principle, this ledger represents an alternative form of "trusted party" that participates in the system. However, unlike the trusted parties proposed in past escrow proposals [30], ledgers do not store any decryption secrets. Moreover, recent advances in consensus protocols, and particularly the deployment of proof-of-work and proof-of-stake cryptocurrency systems. *e.g.,* [17,28,40,52], provide evidence that these ledgers can be operated safely at large scale.

Following the approach outlined by Choudhuri *et al.* [25], we make use of the ledger to *conditionally encrypt* messages such that decryption is only possible following the verifiable publication of the transparency function evaluated over a warrant on the global ledger. For some forms of general purpose ledgers that we seek to use in our system, this can be accomplished using extractable witness encryption (EWE) [18].[4] EWE schemes allow a sender to encrypt under a statement such that decryption is possible only if the decryptor knows of a witness ω that proves that the statement is in some language L, where L parameterizes the scheme. While candidate schemes for witness encryption are known for specific languages (*e.g.* hash proof systems [26,39]), EWE for general languages is unlikely to exist [38].

Building Retrospective ARLEAS from EWE. Our retrospective ARLEAS construction assumes the existence of a global ledger that produces verification proofs π that a warrant has been published to a ledger. As mentioned before, we aim to condition law enforcement access on the issuance of a valid warrant and the publication of a detectable artifact. In a sense, we want to use this published detectable artifact as a key to decrypt relevant ciphertexts. Thus, in this construction, a sender encrypts each message under a statement with a witness that shows evidence that these conditions have been met. This language reasons over (1) the warrant transparency function, (2) a function determining the relevance of the warrant to ciphertext, (3) the global policy function, (4) the judge's warrant approval mechanism, and (5) the ledger's proof of publication function.

On the Requirement of EWE. We justify the use of EWE in our construction by showing that the existence of a secure protocol realizing retrospective ARLEAS implies the existence of a secure EWE scheme for a related language that is deeply linked to the ARLEAS protocol. Intuitively, the witness for this

[4] Using the weaker witness encryption primitive may be possible if the ledger produces *unique* proofs of publication.

language should serve as proof that the protocol has been correctly executed; law enforcement should be able to learn information about a message if and only if the accountability and detectability mechanisms have been run. For the concrete instantiation of retrospective ARLEAS, we give in Sect. 6, this would include getting a valid proof of publication from the ledger. If the protocol is realized with a different accountability mechanism, the witness encryption language will reason over that functionality. No matter the details of the accountability mechanism, we note that it should be difficult for law enforcement to locally simulate the mechanism. If it were computationally feasible, then law enforcement would be able to circumvent the accountability mechanism with ease.

1.3 Contextualizing ARLEAS In The Encryption Debate

This work is motivated by the active global debate on whether to mandate law enforcement access to encrypted communication systems via key escrow. Reduced to its essentials, this debate incorporates two broad sub-questions. First: can mandatory key escrow be deployed safely? Secondly, if the answer to the first question is positive: *should it be deployed?*

We do not seek to address the second question in this work. Many scholars in the policy and technical communities have made significant efforts in tackling this issue [1,11,33,62] and we do not believe that the current work can make a substantial additional contribution. We stress, therefore, that our goal in this work is not to propose techniques for real-world deployment. Numerous practical questions and technical optimizations would need to be considered before ARLEAS could be deployed in practice.

Instead, the purpose of this work is to provide data to help policymakers address the first question. We have observed a growing consensus among stakeholders that key escrow systems should provide strong guarantees of information security as a precondition for deployment. Some stakeholders in the law-enforcement and national security communities grant that key escrow systems *should not be deployed* unless they can mitigate the risk of mass-surveillance via system abuse or compromise.[5] Unfortunately, there is no agreement on the definition of safety, and the technical community remains divided on whether traditional key escrow security measures (such as the use of secure hardware, threshold cryptography and policy safeguards) will be sufficient. We believe that the research community can help to provide answer these questions, and a failure to do so will increase the risk of unsound policy.

Our contribution in this paper is therefore to take a first step towards this goal. We attempt to formalize a notion of abuse-resilient key escrow, and to

[5] For evidence of this consensus, see *e.g.,* the 2018 National Academies of Sciences Report [62], which provides a framework for discussing such questions. See also a recent report by the Carnegie Endowment [33] which chooses to focus only on the problem of escrow for physical devices rather than data in motion, providing the following explanation: "it is much harder to identify a potential solution to the problems identified regarding data in motion in a way that achieves a good balance" (p. 10).

determine whether it can be realized using modern cryptographic techniques. Our work is focused on *feasibility*. With this perspective in mind, we believe that our work makes at least three necessary contributions to the current policy debate:

Surface the notion of cryptographic abuse-resistance. We raise the question of whether key escrow can be made *abuse resistant* using modern cryptographic technologies, and investigate what such a notion would imply. A key aspect of this discussion is the question of detectability: by making abuse and key exfiltration publicly detectable, we can test law enforcement's belief that backdoor secrets can remain secure, and renew security by efficiently re-keying the system.

Separate the problems of prospective and retrospective surveillance. By emphasizing the technical distinctions between prospective and retrospective surveillance, we are able to highlight the design space in which it is realistic to discuss law enforcement access mechanisms. In particular, our technical results in this work illustrate the cryptographic implausibility of retrospective ARLEAS: this may indicate that retrospective surveillance systems are innately susceptible to abuse.

Shift focus to public policy. In defining and providing constructions for prospective and retrospective ARLEAS, we formalize the notion of a global policy function and a transparency function (see Sect. 3). By making these functions explicit, we hope to highlight the difficult policy issues that must be solved before deploying any access mechanism. As noted by Feigenbaum and Weitzner [35], there are limits what cryptography can contribute to this debate; legal and policy experts must do a better job reducing the gray area between rules and principles so that technical requirements can be better specified.

Finally, we note that the existence of a cryptographic construction for ARLEAS may not be sufficient to satisfy lawn enforcement needs. The mathematics for cryptographically strong encryption systems is already public and widespread, and determined criminals may simply implement their own secure messaging systems [32]. Alternatively, they may use steganography or pre-encrypt their messages with strong encryption to prevent "real" plaintext from being recovered by law enforcement while still allowing contacts to read messages [47]. These practical problems will likely limit the power of any ARLEAS and must be considered carefully by policy makers before pushing for deployment.

2 Related Work

The past decade has seen the start of academic work investigating the notion of accountability for government searches. Bates *et al.* [12] focus specifically on CALEA wiretaps and ensuring that auditors can ensure law enforcement compliance with court orders. In the direct aftermath of the Snowden leaks, Segal *et al.* [70] explored how governments could accountably execute searches

without resorting to dragnet surveillance. Liu *et al.* [57] focus on making the number of searches more transparent, to allow democratic processes to balance social welfare and individual privacy. Kroll *et al.* [53,54] investigate different accountability mechanisms for key escrow systems, but stop short of addressing end-to-end encryption systems and the collusion problems we address in this work. Kamara [50] investigates cryptographic means of restructuring the NSA's metadata program. Backes considered anonymous accountable access control [7], while Goldwasser and Park [42] investigate similar notions with the limitation that policies themselves may be secret, due to national security concerns. Frankle *et al.* [37] make use of ledgers to get accountability for search procedures, but their solution cannot be extended to the end-to-end encryption setting. Wright and Varia [79] give a construction that uses cryptographic puzzles to impose a high cost for law enforcement to decrypt messages. Servan-Schreiber and Wheeler [72] give a construction for accountability that randomly selects custodians that law enforcement must access to decrypt a message. Panwar *et al.* [65] attempt to integrate the accountability systems closely with ledgers, but do not use the ledgers to address access to encryption systems. Finally, Scafuro [69] proposes a closely related concept of "break-glass encryption" and give a construction that relies on trusted hardware.

3 Definitions

Notation. Let λ be an adjustable security parameter and $\mathsf{negl}(\lambda)$ be a negligible function in λ. We use $\|$ to denote concatenation, $\overset{c}{\approx}$ to denote computational indistinguishability, and $\overset{s}{\approx}$ to denote statistical indistinguishability. We will write $x \leftarrow \mathsf{Algo}(\cdot)$ to say that x is a specific output of running the algorithm Algo on specific inputs and will write $x \in \mathsf{Algo}(\cdot)$ to indicate that x is an element in the output distribution of Algo, when run with honest random coins. We write $\mathsf{Algo}^{\mathsf{Par}}$ to say that the algorithm Algo is parameterized by the algorithm Par.

Defining ARLEAS. We now formally define the notion of an Abuse-Resistant Law Enforcement Access System (ARLEAS). An ARLEAS is a form of message transmission scheme that supports accountable access by law enforcement officials. To emphasize the core functionality, we base our security definitions on the UC Secure Message Transmission ($\mathcal{F}_{\mathsf{SMT}}$) notion originally introduced by Canetti [21]. Indeed, our systems can be viewed as an extension of a multi-message SMT functionality [22], with added escrow capability.

Parties and System Parameters. An ARLEAS is an interactive message transmission protocol run between several parties and network components:

– User P_i: Users are the primary consumer of the end-to-end encrypted service or application. These parties, which may be numerous, interact with the system by sending messages to other users.

- Judge P_J: The judge is responsible for determining the validity of a search and issuing search warrants to law enforcement. The judge interacts with the system by receiving warrant requests and choosing to deny or approve the request.
- LawEnforcement P_{LE}: Law enforcement is responsible for conducting searches pursuant to valid warrants authorized by a judge. Law enforcement interacts with the system by requesting warrants from the judge and collecting the plaintext messages relevant to their investigations.

A concrete ARLEAS system also assumes the existence of a communication network that parties can use to transmit encrypted messages to other users. To support law enforcement access, it must be possible for law enforcement to "tap" this network and receive encrypted communications between targeted users. For the purposes of this exposition, we will assume that law enforcement agents have access to any communications transmitted over the network (*i.e.,* the network operates as a transparent channel.) In practice, a service provider would handle the transmissions of ciphertexts. This service provider would also be responsible for storing ciphertext and metadata, and providing this information to law enforcement. Our simplified model captures the worst case network security assumption, where the service provider cooperates with all law enforcement requests. Service providers would also be responsible for checking that messages sent by users are compliant with the law enforcement access protocol. We move this responsibility to the receiver for simplicity. We discuss the role of service providers in more detail in the full version.

An ARLEAS system is additionally parameterized by four functions, which are selected during a trusted setup phase:

- $t(w)$: the deterministic *transparency function* takes as input a warrant w and outputs specific information about the warrant that can be published to the general public.
- $p(w)$: the deterministic *global policy* function takes as input a warrant w and outputs 1 if this warrant is allowed by the system.
- $\theta(w, \mathsf{meta})$: the deterministic *warrant scope check* takes as input a warrant w and per-message metadata meta. It outputs 1 if meta is in scope of w for surveillance.
- $v(\mathsf{meta}, aux)$: The deterministic *metadata verification functionality* takes as input metadata associated with some message meta and some auxiliary information aux and determines if the metadata is correct. This auxiliary information could contain the ciphertext, global timing information, or some authenticated side channel information.

We discuss concrete instantiations of these functions in the full version of the paper.

ARLEAS Scheme. An ARLEAS scheme comprises a set of six possibly interactive protocols. We provide a complete API specification for these protocols in later sections:

- Setup. On input a security parameter, this trusted setup routine generates all necessary parameters and keys needed to run the full system.
- SendMessage. On input a message m, metadata meta, and a recipient identity, this protocol sends an encrypted message from one party to another.
- RequestWarrant. On input a description of the warrant request, this procedure allows law enforcement to produce a valid warrant.
- ActivateWarrant. Given a warrant w, this protocol allows law enforcement and a judge to confirm and activate a warrant.
- VerifyWarrantStatus. Given a warrant w, this protocol is used to verify that a warrant is valid and active.
- AccessMessage. In the retrospective case, this protocol is used by law enforcement to open a message.

UC Ideal Functionality. To define the properties of an ARLEAS system, we present a formal UC ideal functionality $\mathcal{F}_{\text{ARLEAS}}$ in Fig. 1. Recalling that ARLEAS can be instantiated in one of two modes, supporting only prospective or retrospective surveillance, we present a single definition that supports a parameter, $\mathsf{mode} \in \{\mathsf{pro}, \mathsf{ret}\}$.

Ideal World. For any ideal-world adversary $\mathcal{S}$ with auxiliary input $z \in \{0,1\}^*$, input vector x, and security parameter λ, we denote the output of the ideal world experiment by $\mathbf{Ideal}_{\mathcal{S},\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{mode}}}(1^\lambda, x, z)$.

Real World. The real world protocol starts with the adversary $\mathcal{A}$ selecting a subset of the parties to compromise $\mathcal{P}^{\mathcal{A}} \subset \mathcal{P}$, where $\mathcal{P}^{\mathcal{A}} \subset \{\{P_i\}, \{P_{\text{LE}}\}, \{P_{\text{LE}}, P_J\}\}$, where we denote sender with P_i and receiver with P_j. We limit the subsets of parties that can be compromised to these cases, because any other combination is trivial to simulate or can be deducted from the other cases. For example, if both P_i and P_j would be corrupted, there is nothing stopping them from not using the system. Moreover, we also don't consider the case where P_J is the only corrupted party, this case is a more specific then when both P_{LE} and P_J are corrupted and P_J on its own doesn't have any additional information to achieve anything different. All parties engage in a real protocol execution Π, the adversary $\mathcal{A}$ sends all messages on behalf of the corrupted parties and can choose any polynomial time strategy.

In a real world protocol we assume that communication between a sender P_i and receiver P_j happens over a transparent channel, meaning all other parties are able to receive all communication. We make this choice to simplify the protocol and security proofs. In the real world, this can be modeled with a service provider relaying messages between P_i and P_j that always complies with law enforcement requests and hands over encrypted messages when presented with a valid warrant. Note that this makes our modeling the worst case scenario, and therefore captures more selective service providers. Additionally, in practice, this service provider would validate if messages are well-formed to make sure P_i and P_j follow the real protocol.

For any adversary $\mathcal{A}$ with auxiliary input $z \in \{0,1\}^*$, input vector x, and security parameter λ, we denote the output of Π by $\mathbf{Real}_{\mathcal{A},\Pi}(1^\lambda, x, z)$.

Functionality $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{mode}}$

The ideal functionality is parameterized by $\mathsf{mode} \in \{\mathsf{pro}, \mathsf{ret}\}$, a metadata verification function $v : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, the transparency function $t(\cdot)$, the global policy function $p(\cdot)$, and the warrant scope check functionality $\theta(\cdot,\cdot)$. The three latter functions are as defined above. We denote the session identifier as sid to separate different runs of the same protocol. We have several parties:

- $P_1, \ldots, P_n$: participants in the system
- P_J: the generator of a warrant
- P_{LE}: Law enforcement that can read the message given a valid warrant

Send Message: Upon receiving a message $(\mathsf{SendMessage}, \mathsf{sid}, P_j, m, \mathsf{meta}, \mathsf{valid})$ where $\mathsf{valid} \in \{0,1\}$ from party P_i, it sends $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta})$ to the adversary. If (sid, c) is received from the adversary,

- If $\mathsf{valid} = 0$ or $v(\mathsf{meta}, aux) = 0$, send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m)$ to P_i and send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, 0)$ to P_{LE}.
- If $\mathsf{valid} = 1$, $v(\mathsf{meta}, aux) = 1$, and there is no entry w in the active warrant table W_{active} send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m)$ to P_i and P_j, and send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c)$ to P_{LE}.
- If $\mathsf{valid} = 1$, $v(\mathsf{meta}, aux) = 1$, and there is an entry w in the active warrant table W_{active} send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m)$ to P_i, P_j, and P_{LE}.

Finally, store $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m)$ in the message table M.

Request Warrant: Upon receiving a message $(\mathsf{RequestWarrant}, \mathsf{sid}, w)$ from P_{LE}, the ideal functionality first checks if $p(w) = 1$, responding with $\perp$ and aborting if not. Otherwise, the ideal functionality sends $(\mathsf{ApproveWarrant}, w)$ to P_J. If P_J responds with $(\mathsf{Disapprove})$, the trusted functionality sends $\perp$ to P_{LE}. If P_J responds with $(\mathsf{Approve})$, the trusted functionality sends $(\mathsf{Approve})$ to P_{LE}, and stores the entry w in the issued warrant table W_{issued}.

Activate Warrant: Upon receiving a message $(\mathsf{ActivateWarrant}, \mathsf{sid}, w)$ from P_{LE}, the ideal functionality checks to see if $w \in W_{\text{issued}}$, responding with $\perp$ and aborting if not. If $w \in W_{\text{issued}}$, the trusted functionality adds the entry w to the active warrant table W_{active}, computes $t(w)$, and sends $(\mathsf{NotifyWarrant}, t(w))$ to all parties and the adversary.

Verify Warrant Status: Upon receiving message $(\mathsf{VerifyWarrantStatus}, \mathsf{sid}, c, \mathsf{meta}, w)$ from P_{LE}, if $\mathsf{mode} = \mathsf{pro}$, the ideal functionality responds with $\perp$ and aborts. Otherwise, if $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m) \in M$ and $w \in W_{\text{active}}$ such that $\theta(w, \mathsf{meta}) = 1$, the ideal functionality returns 1. Finally, if $\theta(w, \mathsf{meta}) = 0$ or $w \notin W_{\text{active}}$, it returns 0.

Access message: Upon receiving message $(\mathsf{AccessData}, \mathsf{sid}, c, \mathsf{meta}, w)$ from P_{LE}, if $\mathsf{mode} = \mathsf{pro}$, the ideal functionality responds with $\perp$ and aborts. Otherwise, if $(\mathsf{Sent}, \mathsf{sid}, \mathsf{meta}, c, m) \in M$ and $w \in W_{\text{active}}$ such that $\theta(w, \mathsf{meta}) = 1$, the ideal functionality returns m. Finally, if $\theta(w, \mathsf{meta}) = 0$, it returns 0.

Fig. 1. Ideal functionality for an Abuse Resistant Law Enforcement Access System.

Protocol $\mathbf{Real}_{\mathcal{A},\Pi}(1^\lambda, x, z)$

$\mathbf{Real}_{\mathcal{A},\Pi}(1^\lambda, x, z)$ is parameterized by the protocol Π = (Setup, SendMessage, RequestWarrant, ActivateWarrant, VerifyWarrantStatus, AccessMessage) and a variable mode $\in$ {pro, ret}.

1. When $\mathbf{Real}_{\mathcal{A},\Pi}(1^\lambda, x, z)$ is initialized, then all parties engage in the interactive protocol Π.Setup
2. When P_i is activated with (SendMessage, sid, $P_j, m, 1$), parties P_i, P_j, and P_{LE} engage in the interactive protocol Π.SendMessage. P_{LE} learns some metadata meta about the message.
3. When P_i is activated with (SendMessage, sid, $P_j, m, 0$), parties P_i, and P_{LE} engage in the interactive protocol Π.SendMessage (with P_j not getting output). P_{LE} learns some metadata meta about the message.
4. When P_{LE} is activated with (RequestWarrant, sid, $\hat{w}$), parties P_{LE} and P_J engage in the interactive protocol Π.RequestWarrant.
5. When P_{LE} is activated with (ActivateWarrant, sid, w), all parties engage in the interactive protocol Π.ActivateWarrant.
6. When P_{LE} is activated with (VerifyWarrantStatus, sid, c, meta, w), if mode = pro, P_{LE} returns $\perp$. Otherwise, P_{LE} calls the non-interactive functionality Π.VerifyWarrantStatus(c, meta, w)
7. When P_{LE} is activated with (AccessData, sid, c, meta, w), if mode = pro, P_{LE} returns $\perp$. Otherwise, P_{LE} calls the non-interactive functionality Π.AccessMessage(c, meta, w)

Fig. 2. The real world experiment for a protocol implementing $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\text{mode}}$

Functionality $\mathcal{L}^{\text{Verify}}$

GetCounter: Upon receiving (GetCounter) from any party, return ℓ.

Post: Upon receiving (Post, x), the trusted party increments ℓ by 1, computes the proof of publication π_{publish} on $(\ell||x)$ such that Verify$((\ell\|x), \pi_{\text{publish}}) = 1$. Add the entry $(\ell, x, \pi_{\text{publish}})$ to the entry table T. Respond with $(\ell, x, \pi_{\text{publish}})$

GetVal: Upon receiving (GetVal, ℓ), check if there is an entry $(\ell, x, \pi_{\text{publish}})$ in the entry table T. If not, return $\perp$. Otherwise, return $(\ell, x, \pi_{\text{publish}})$.

Fig. 3. Ideal functionality for a proof-of-publication ledger, from [25].

Definition 1. *A protocol Π is said to be a secure ARLEAS protocol computing $\mathcal{F}_{ARLEAS}^{v,t,p,\theta,mode}$ if for every PPT real-world adversary $\mathcal{A}$, there exists an ideal-world PPT adversary $\mathcal{S}$ corrupting the same parties such that for every input x and auxiliary input z it holds that*

$$\mathbf{Ideal}_{\mathcal{S},\mathcal{F}_{ARLEAS}^{v,t,p,\theta,mode}}(1^\lambda, x, z) \overset{c}{\approx} \mathbf{Real}_{\mathcal{A},\Pi}(1^\lambda, x, z)$$

$\pi_{PRO}^{v,t,p,\theta}.\mathsf{Setup}$:

- All users send (CRS) to $\mathcal{F}_{CRS}^{\Pi_{\mathrm{NIZK}}.\mathsf{ZKSetup}}$ to retrieve the common reference string for the NIZK scheme and all users send (CRS) to $\mathcal{F}_{CRS}^{\Pi_{\mathrm{NISC}}.\mathsf{GenCRS}}$ to retrieve the common reference string for the NISC scheme $\mathsf{CRS}_{\mathsf{NISC}}$.
- Each user P_j computes $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \Pi_{\mathrm{Enc}}.\mathsf{KeyGen}(1^\lambda)$ and sends pk_j to P_{LE} and to each P_i via $\mathcal{F}_{AUTH}$.
- The judge P_J computes $(\mathsf{pk}_{\mathrm{sign}}, \mathsf{sk}_{\mathrm{sign}}) \leftarrow \Pi_{\mathrm{Sign}}.\mathsf{KeyGen}(1^\lambda)$ and send $\mathsf{pk}_{\mathrm{sign}}$ to all other users via $\mathcal{F}_{AUTH}$.
- Law enforcement P_{LE} runs $\pi_{\mathrm{PRO}}^{v,t,p,\theta}.\mathsf{ActivateWarrant}$ with an empty set $\emptyset$ as the valid warrants.

$\pi_{PRO}^{v,t,p,\theta}.\mathsf{SendMessage}$:

- The sender P_i computes the ciphertext $(c_1, c_2, \pi, \mathsf{meta})$ as follows, and sends it to P_j and P_{LE} via $\mathcal{F}_{AUTH}$:
  - Send (GetCounter) to $\mathcal{L}^{\mathsf{Verify}}$ and receive the current counter ℓ. Then query $\mathcal{L}^{\mathsf{Verify}}$ on (GetVal, ℓ) to receive the latest posting $(\ell, x, \pi_{\mathsf{publish}})$. Parse x as $(\mathsf{nisc}_1^{\mathrm{public}}, \pi, \mathsf{info})$. If $\Pi_{\mathrm{NIZK}}.\mathsf{ZKVerify}(\mathsf{nisc}_1^{\mathrm{public}}, \mathsf{info}, \pi) = 0$ or $\mathcal{L}^{\mathsf{Verify}}.\mathsf{Verify}(\ell \| (\mathsf{nisc}_1^{\mathrm{public}}, \pi, \mathsf{info}), \pi_{\mathsf{publish}}) = 0$ return $\perp$ and halt.
  - $c_1 \leftarrow \Pi_{\mathrm{Enc}}.\mathsf{Enc}(\mathsf{pk}_j, m; r_1)$, where $r_1 \xleftarrow{\$} \{0,1\}^\lambda$
  - Create meta
  - $\mathsf{nisc}_2 \leftarrow \Pi_{\mathrm{NISC}}.\mathsf{NISC}_2(\mathsf{CRS}_{\mathsf{NISC}}, I_{|\mathsf{info}|}, (m, \mathsf{meta}), \mathsf{nisc}_1^{\mathrm{public}}; r_2)$, where $r_2 \xleftarrow{\$} \{0,1\}^\lambda$
  - $c_2 \leftarrow \mathsf{nisc}_2$
  - Use $\Pi_{\mathrm{NIZK}}.\mathsf{ZKProve}$ to compute π such that

$$\pi \leftarrow NIZK\left\{(m, r_1, r_2) : \begin{array}{r} c_1 = \Pi_{\mathrm{Enc}}.\mathsf{Enc}(\mathsf{pk}_j, m; r_1) \wedge \\ c_2 = \Pi_{\mathrm{NISC}}.\mathsf{NISC}_2(\mathsf{CRS}_{\mathsf{NISC}}, I_{|\mathsf{info}|}, (m, \mathsf{meta}), \mathsf{nisc}_1^{\mathrm{public}}; r_2) \end{array}\right\}$$

- Upon receiving c from P_i, P_j calls $\pi_{\mathrm{PRO}}^{v,t,p,\theta}.\mathsf{VerifyMessage}$ on c. If the output is 1, then recover the message as $m \leftarrow \Pi_{\mathrm{Enc}}.\mathsf{Dec}(\mathsf{sk}_j, c_2)$
- Upon receiving c from P_i, P_{LE} calls $\pi_{\mathrm{PRO}}^{v,t,p,\theta}.\mathsf{VerifyMessage}$ on c. If the output is 1, then recover the message as $m \leftarrow \Pi_{\mathrm{NISC}}.\mathsf{Evaluate}(\mathsf{CRS}_{\mathsf{NISC}}, \mathsf{nisc}_2, \mathsf{nisc}_1^{\mathrm{private}})$

$\pi_{PRO}^{v,t,p,\theta}.\mathsf{VerifyMessage}$:

- Any party parses $(c_1, c_2, \pi, \mathsf{meta}) \leftarrow c$ and verifies that π is correct and computes $v(\mathsf{meta}, aux)$, aborting if the output is 0. Otherwise, output 1.

$\pi_{PRO}^{v,t,p,\theta}.\mathsf{RequestWarrant}$:

- P_{LE} sends (RequestWarrant, $\hat{w}$) to P_J via $\mathcal{F}_{AUTH}$. P_J then either decides to send (Disapprove) to P_{LE} and halt or executes the following:
  - Verify that $p(\hat{w}) = 1$. If not send (Disapprove) to P_{LE} and abort.
  - $\sigma \leftarrow \Pi_{\mathrm{Sign}}.\mathsf{Sign}(\mathsf{sk}_{\mathrm{sign}}, \hat{w})$
  - Send the signed warrant $w = (\hat{w}, \sigma)$ to P_{LE} via $\mathcal{F}_{AUTH}$.

$\pi_{PRO}^{v,t,p,\theta}.\mathsf{ActivateWarrant}$:

- P_{LE} adds the new warrant w to the set of valid warrants $\mathcal{W}$. Let $w^* = w_1 \| \ldots \| w_{|\mathcal{W}|}$ for $w_i = (\hat{w}_i, \sigma_i) \in \mathcal{W}$.
- $(\mathsf{nisc}_1^{\mathrm{public}}, \mathsf{nisc}_1^{\mathrm{private}}) \leftarrow \Pi_{\mathrm{NISC}}.\mathsf{NISC}_1(\mathsf{CRS}_{\mathsf{NISC}}, w^*; r)$ and record $\mathsf{nisc}_1^{\mathrm{private}}$
- Compute $\mathsf{info} \leftarrow \{t(w) | w \in \mathcal{W}\}$
- Use $\Pi_{\mathrm{NIZK}}.\mathsf{ZKProve}$ to compute π such that

$$\pi \leftarrow NIZK\left\{(\mathcal{W}, \mathsf{nisc}_1^{\mathrm{private}}, r) : \begin{array}{r} \mathsf{info} = \{t(w) | w \in \mathcal{W}\} \wedge \\ (\mathsf{nisc}_1^{\mathrm{public}}, \mathsf{nisc}_1^{\mathrm{private}}) \leftarrow \Pi_{\mathrm{NISC}}.\mathsf{NISC}_1(\mathsf{CRS}_{\mathsf{NISC}}, w^*; r) \wedge \\ \forall (\hat{w}, \sigma) \in \mathcal{W}, \Pi_{\mathrm{Sign}}.\mathsf{Verify}(\mathsf{pk}_{\mathrm{sign}}, \hat{w}, \sigma) = p(\hat{w}) = 1 \end{array}\right\}$$

- Send $(\mathsf{Post}, (\mathsf{nisc}_1^{\mathrm{public}}, \pi, \mathsf{info}))$ to $\mathcal{L}^{\mathsf{Verify}}$ and receive $(\ell, x, \pi_{\mathsf{publish}})$.

Fig. 4. Our construction of a protocol $\pi_{\mathrm{PRO}}^{v,t,p,\theta}$ that UC-realizes $\mathcal{F}_{\mathrm{ARLEAS}}^{v,t,p,\theta,\mathsf{pro}}$

4 Building Blocks

Proof-of-Publication Ledgers. Our work makes use of a public append-only ledger that can produce a publicly-verifiable *proof of publication*. This concept was formalized by Goyal *et al.* [45], Choudhuri *et al.* [25], and Kaptchuk *et al.* [51], but related ideas have also been previously used by Liu *et al.* to realize time-lock encryption [58]. Plausible candidates for such ledgers have been the subject of great interest, due to the deployment of blockchains and other consensus networks [59]. Significant work has been done to formalize the notion of a public, append-only ledger [8,9,24,45] and study its applications to cryptographic protocols [3,15,25]. This work uses a simplified ledger interface formalized in [25] that abstracts away details such as timing information and temporary inconsistent views that are modeled in [9]. However, this simplified view captures the *eventual* functionality of the complex models, and is therefore equivalent for our purposes (Fig. 2).

The ledger ideal functionality is provided in Fig. 3. This functionality allows users to post arbitrary information to the ledger; this data is associated with a particular index on the ledger, with which any user can retrieve the original data as well as a proof of publication. For security, our functionality encodes a notion we refer to as *ledger unforgeability*, which requires that there exists an algorithm to verify a proof that a message has been posted to the ledger, and that adversaries cannot forge this proof.

Authenticated Communication. We use a variant of Canetti's ideal functionality for authenticated communication, $\mathcal{F}_{AUTH}$, to abstract the notion of message authentication [21]. Due to space constraints, we omit the ideal functionality in this shortened version. Since we restrict our analysis to static corruption, we simplify this functionality to remove the adaptive corruption interface.[6]

Simulation Extractable Non-interactive Zero Knowledge. In our protocols we require non-interactive zero knowledge proofs of knowledge that are simulation extractable. To preserve space, we refer the reader to the definitions of Sahai [67] and De Santis *et al.* [29]. Rather than rely on UC functionalities, we employ a NIZK directly in our protocols.

Multi-sender Non-interactive Secure Computation. When instantiating our prospective protocol for arbitrary predicates in Sect. 5, we will require the use of Non-interactive Secure Computation (NISC) [2,49]. In NISC, a receiver can post an encryption embedding a secret x_1 such that senders with secret x_2 can reveal $f(x_1, x_2)$ to the receiver by sending only a single message. Realizing such a scheme (see [49]) is feasible in the CRS model [21,23] from two-round, UC-secure malicious oblivious transfer [27,63], Yao's garbled circuits [48], and generic non-interactive zero knowledge (see Sect. 4). The resulting protocols,

[6] Note that this ideal functionality only handles a single message transfer, but to achieve multiple messages, we rely on universal composition and use multiple instances of the functionality.

however, are very inefficient and require non-blackbox use of the underlying cryptographic primitives. While this is sufficient for our purposes, we note that depending on specific functionality required in an instantiation of ARLEAS, it may be possible to use more efficient constructions (*i.e.* depending on the size of the predicate circuit, etc.) Because the notation for NISC protocols varies, we fix it for this work below. We omit the ideal functionality of multi-sender NISC from [2], due to space constraints. Because we require non-blackbox use of the primitive, we will use it directly rather than as a hybrid.

Definition 2 (Multi-sender Non-interactive Secure Computation). *A garbling scheme for a functionality* $f : \{0,1\}^{\mathsf{input}_1} \times \{0,1\}^{\mathsf{input}_2} \rightarrow \{0,1\}^{\mathsf{output}}$ *is a tuple of PPT algorithms* $\Pi_{NISC} := (\mathsf{GenCRS}, \mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{Evaluate})$ *such that*

- $\mathsf{GenCRS}(1^\lambda, \mathsf{input}; r) \rightarrow (\mathsf{CRS}_{\mathsf{NISC}}, \tau_{\mathsf{NISC}})$*:* GenCRS *takes the security parameter* 1^n *and outputs a CRS, along with a simulation backdoor* τ_{NISC}*. When we explicitly need to specify the randomness, we will include it as* r *as here.*
- $\mathsf{NISC}_1(\mathsf{CRS}_{\mathsf{NISC}}, x_1; r) \rightarrow (\mathsf{nisc}_1^{public}, \mathsf{nisc}_1^{private})$*:* NISC_1 *takes in the CRS and an input* $x \in \{0,1\}^{\mathsf{input}_1}$ *and outputs the first message* NISC_1*. When we explicitly need to specify the randomness, we will include it as* r *as here.*
- $\mathsf{NISC}_2(\mathsf{CRS}_{\mathsf{NISC}}, f, x_2, \mathsf{nisc}_1^{public}; r) \rightarrow \mathsf{nisc}_2$*:* NISC_2 *takes in the CRS, a circuit* C*, an input* $x_2 \in \{0,1\}^{\mathsf{input}_2}$ *and the first garbled circuit message* nisc_1^{public}*. It outputs the second message* nisc_2*. When we explicitly need to specify the randomness, we will include it as* r *as here.*
- $\mathsf{Evaluate}(\mathsf{CRS}_{\mathsf{NISC}}, \mathsf{nisc}_2, \mathsf{nisc}_1^{private})$*:* $\mathsf{Evaluate}$ *takes as input the second GC message* nisc_2 *along with the private information* $\mathsf{nisc}_1^{private}$ *and outputs* $y \in \{0,1\}^{\mathsf{output}}$ *or the error symbol* $\perp$

We omit the ideal world security definition for multi-sender NISC, due to space constraints. It can be found in [2].

Witness Encryption and Extractable Witness Encryption. Our *retrospective* constructions require extractable witness encryption (EWE) [18], a variant of witness encryption in which the existence of a distinguisher can be used to construct an extractor for the necessary witness [41]. While EWE is a strong assumption, in later sections of this work we show that it is a minimal requirement for the existence of retrospective ARLEAS.

To preserve space we give the formal definition in the full version of our paper and we describe it informally here. An extractable witness encryption scheme is parameterized by an NP-language L and has two algorithms $\Pi_{\mathrm{EWE}} = (\mathsf{Enc}, \mathsf{Dec})$. Encryption uses a statement x to encrypt a plaintext message m, while decryption uses a witness ω such that $(x, \omega) \in \mathcal{R}_L$ to recover the plaintext.

This scheme has two properties: correctness and extractable security. Correctness implies decryption recovers the plaintext if the witness is valid. Extractable security says that if an adversary can distinguish between two encrypted messages, there exists an extractor that can extract the witness of the statement.

$\pi_{RET}^{v,t,p,\theta}$.Setup:

- All users send (CRS) to $\mathcal{F}_{CRS}^{\Pi_{\text{NIZK}}.\mathsf{ZKSetup}}$ to retrieve the common reference string for the NIZK scheme.
- P_J computes $(\mathsf{pk}_{\text{sign}}, \mathsf{sk}_{\text{sign}}) \leftarrow \Pi_{\text{Sign}}.\mathsf{KeyGen}(1^\lambda)$ and sends $\mathsf{pk}_{\text{sign}}$ to all other users via $\mathcal{F}_{AUTH}$.

$\pi_{RET}^{v,t,p,\theta}$.SendMessage:

- The sender P_i computes the ciphertext $(c_1, c_2, c_3, \pi, \mathsf{meta})$ as follows, and sends it to P_j and P_{LE} via $\mathcal{F}_{AUTH}$:
  - Sample $r \leftarrow \{0,1\}^\lambda$
  - Query the random oracle to obtain the hashes:
    $(\mathsf{HashConfirm}, r_1) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“ENC”} \| r \| m))$,
    $(\mathsf{HashConfirm}, r_2) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“WE”} \| r \| m))$, and
    $(\mathsf{HashConfirm}, r_3) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“RP”} \| r))$
  - $c_1 \leftarrow \Pi_{\text{Enc}}.\mathsf{Enc}(\mathsf{pk}, r; r_1)$, $c_2 \leftarrow \Pi_{\text{EWE}}.\mathsf{Enc}(\mathsf{meta}, r; r_2)$, and $c_3 \leftarrow m \oplus r_3$
  - Use $\Pi_{\text{NIZK}}.\mathsf{ZKProve}$ to compute $\pi \leftarrow NIZK\{(r, r_1, r_2) : c_1 = \Pi_{\text{Enc}}.\mathsf{Enc}(\mathsf{pk}_j, r; r_1) \wedge c_2 = \Pi_{\text{EWE}}.\mathsf{Enc}(\mathsf{meta}, r; r_2)\}$
- Upon receiving (send, c), P_j performs the following steps:
  - Call $\pi_{\text{RET}}^{v,t,p,\theta}.\mathsf{VerifyMessage}$ on c, aborting if the output is 0;
  - Compute $r' \leftarrow \Pi_{\text{Enc}}.\mathsf{Dec}(\mathsf{sk}_j, c_1)$
  - $(\mathsf{HashConfirm}, r_3) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“RP”} \| r'))$
  - Compute $m' \leftarrow c_3 \oplus r_3$
  - $(\mathsf{HashConfirm}, r_1) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“ENC”} \| r' \| m'))$
  - $(\mathsf{HashConfirm}, r_2) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“WE”} \| r' \| m'))$
  - Then to verify that the message has not been mauled, P_j recomputes $c_1' \leftarrow \Pi_{\text{Enc}}.\mathsf{Enc}(\mathsf{pk}_j, r'; r_1)$ and $c_2' \leftarrow \Pi_{\text{EWE}}.\mathsf{Enc}(\mathsf{meta}, r'; r_2)$. If $c_1 \neq c_1'$ or $c_2 \neq c_2'$, return $\perp$. Otherwise, return m'.
- Upon receiving (send, c), P_{LE} calls $\pi_{\text{RET}}^{v,t,p,\theta}.\mathsf{VerifyMessage}$ on c, aborting if the output is 0, and then calls $\pi_{\text{RET}}^{v,t,p,\theta}.\mathsf{AccessMessage}$ on c.

$\pi_{RET}^{v,t,p,\theta}$.VerifyMessage :

- Any party parses $(c_1, c_2, c_3, \pi, \mathsf{meta}) \leftarrow c$ and verifies that π is correct and computes $v(\mathsf{meta}, aux)$, aborting if the output is 0. Otherwise, output 1.

$\pi_{RET}^{v,t,p,\theta}$.RequestWarrant:

- P_{LE} sends $(\mathsf{RequestWarrant}, \hat{w})$ to P_J via $\mathcal{F}_{AUTH}$. P_J then either decides to send $(\mathsf{Disapprove})$ to P_{LE} and halt or executes the following:
  - Verify that $p(\hat{w}) = 1$. If not send $(\mathsf{Disapprove})$ to P_{LE} and abort.
  - $\sigma \leftarrow \Pi_{\text{Sign}}.\mathsf{Sign}(wsk, \hat{w})$
  - Send the signed warrant $w = (\hat{w}, \sigma)$ to P_{LE} via $\mathcal{F}_{AUTH}$.

$\pi_{RET}^{v,t,p,\theta}$.ActivateWarrant:

- P_{LE} computes $\mathsf{info} \leftarrow t(w)$; uses $\Pi_{\text{NIZK}}.\mathsf{ZKProve}$ to compute $\pi \leftarrow NIZK\{(w) : w = (\hat{w}, \sigma), \Pi_{\text{Sign}}.\mathsf{Verify}(\mathsf{pk}_{\text{sign}}, \hat{w}, \sigma) = 1 \wedge \mathsf{info} \leftarrow t(w)\}$; and sends $(\mathsf{Post}, (\mathsf{info}, \pi))$ to $\mathcal{L}^{\mathsf{Verify}}$. It receives and returns $(\ell, \mathsf{info}, \pi_{\mathsf{publish}})$.

$\pi_{RET}^{v,t,p,\theta}$.VerifyWarrantStatus:

- P_{LE} calls $\Pi_{\text{EWE}}.\mathsf{Dec}(c_2, \mathsf{meta}, (\hat{w}, \sigma), (\ell, \mathsf{info}, \pi_{\mathsf{publish}}))$. If the output is $\perp$, return 0. Otherwise, return 1.

$\pi_{RET}^{v,t,p,\theta}$.AccessMessage:

- P_{LE} computes $r' \leftarrow \Pi_{\text{EWE}}.\mathsf{Dec}(c_2, \mathsf{meta}, (\hat{w}, \sigma), (\ell, \mathsf{info}, \pi_{\mathsf{publish}}))$.
- $(\mathsf{HashConfirm}, r_3) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“RP”} \| r'))$
- Recovers $m' \leftarrow c_3 \oplus r_3$.
- $(\mathsf{HashConfirm}, r_1) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“ENC”} \| r' \| m'))$
- $(\mathsf{HashConfirm}, r_2) \leftarrow \mathcal{G}_{\mathsf{pRO}}(\mathsf{HashQuery}, (\text{“WE”} \| r' \| m'))$
- Recomputes $c_1' \leftarrow \Pi_{\text{Enc}}.\mathsf{Enc}(\mathsf{pk}_j, r'; r_1)$ and $c_2' \leftarrow \Pi_{\text{EWE}}.\mathsf{Enc}(\mathsf{meta}, r'; r_2)$. If $c_1' = c_1$ and $c_2' = c_2$, P_{LE} returns m' and $\perp$ otherwise.

Fig. 5. Our construction of a protocol $\pi_{\text{RET}}^{v,t,p,\theta}$ that UC-realizes $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{ret}}$

Programmable Global Random Oracle Model. The security proof for our retrospective construction makes use of the programmable global random oracle model, introduced in [20]. We omit the ideal functionality $\mathcal{G}_{\mathsf{pRO}}$ from [20] due to space constraints.

5 Prospective Solution

In this section we describe a prospective ARLEAS scheme, which supports arbitrary predicates. Recall that the key feature of the prospective case is that warrants must be activated *before* targets perform encryption. A key implication of this setting is that new cryptographic material can be generated and distributed to users each time law enforcement updates the set of active warrants. The technical challenge, therefore, is to ensure that this material is distributed in such a way that the surveillance it permits is *accountable*, without revealing to targets any confidential information about which messages are being accessed.

For generality, our main construction supports targeting by allowing warrants to specify an arbitrary predicate over the *metadata* of a transmitted messages. In practice, we realize this functionality through the use of public ledgers and non-interactive secure computation techniques.

5.1 UC-Realizing $\mathcal{F}_{\mathbf{ARLEAS}}^{v,t,p,\theta,\mathbf{pro}}$ for Arbitrary Predicates

To realize prospective ARLEAS, each user must encrypt each message in two separate forms. The first ciphertext uses standard PKE ciphertext to encrypt the message directly to the recipient, as is standard in many end-to-end encrypted messaging systems. The second ciphertext represents a "law enforcement access field" that is designed to permit authorized surveillance. To construct the second ciphertext, we require a mechanism that enables law enforcement access if and only if the warrant is active and valid for the specific message metadata being transmitted. To ensure that the transmission is consistent (*i.e.,* the plaintexts contained in each ciphertext is the same), the two ciphertexts are bound together by using non-interactive zero knowledge proof of knowledge that can be verified by all parties in the system.

Our construction relies on non-interactive secure computation (NISC) [49]. Recall that a NISC scheme for some function f allows a receiver to post an encryption of some secret x_1 such that all players can reveal $f(x_1, x_2)$ to the receiver with only one message, without revealing anything about x_2 beyond the output of the function. For the following construction, we require a NISC scheme for the function I_k, defined as $I_k((w_1, w_2, \ldots, w_k), (m, \mathsf{meta})) = m \wedge (\theta(\mathsf{meta}, w_1) \vee \ldots \vee \theta(\mathsf{meta}, w_k))$.

This function evaluates the warrant scope check functionality on the metadata over k different warrants. If any of them evaluate to true, the message is output. Otherwise, I_k outputs 0. Note that the number of warrants is an explicit parameter of the function and its circuit representation.

Law enforcement begins by posting the first message of the NISC scheme, embedding as input their k warrants, along with the transparency information and proof of correctness. Senders generate and send the ciphertext (c_1, c_2, π), generated as follows. c_1 remains a normal public key ciphertext for the recipient. c_2 is the second message of the NISC scheme, for the function I_k and embedding the inputs m, meta. Most known realizations of NISC rely on garbled circuits, with the second message containing the garbling of the intended function and hardcoding the sender's inputs. π is a zero-knowledge proof demonstrating that the two ciphertexts contain the same message and that they were each generated correctly with respect to the first message of the NISC.

Upon receiving the resulting ciphertext, law enforcement can attempt to decrypt by evaluating the NISC ciphertext. By the security of the NISC scheme, law enforcement will only learn information about the plaintext if they have a relevant warrant and posted the required transparency information, accomplishing our goal.

We give a description of the prospective ARLEAS protocol $\pi_{\mathrm{PRO}}^{v,t,p,\theta}$ in Fig. 4.

Theorem 1. *Assuming a CCA secure public key encryption scheme Π_{Enc}, a SUF-CMA secure signature scheme Π_{Sign}, a NIZK scheme Π_{NIZK}, and an NISC scheme Π_{NISC}, $\pi_{PRO}^{v,t,p,\theta}$ (presented in Fig. 4) UC-realizes $\mathcal{F}_{ARLEAS}^{v,t,p,\theta,pro}$ initialized in prospective mode in the $\mathcal{L}^{\mathsf{Verify}}$, $\mathcal{F}_{CRS}^{\Pi_{NIZK}.\mathsf{ZKSetup}}$, $\mathcal{F}_{CRS}^{\Pi_{NISC}.\mathsf{GenCRS}}$, $\mathcal{F}_{AUTH}-$hybrid model.*

Security Proof. We give the security proof in the full version of the paper. The simulator is straight forward, taking advantage of the NIZKs and the NISC to facilitate extraction. The proof first simulates just a user, then law enforcement, and then both the judge and law enforcement.

6 Retrospective Solution

In the previous section we proposed a protocol to realize ARLEAS under the restriction that access would be prospective only. That protocol requires that law enforcement must activate a warrant and post the resulting parameters on the ledger before any targeted communication occurs. In this section we address the retrospective case. The key difference in this protocol is that law enforcement may activate a warrant at any stage of the protocol, even after a target communication has occurred.

In this setting we assume law enforcement has a way of getting messages that were sent in the past. As described before, we take the simplifying assumption that messages automatically get send to law enforcement. In practice, either a service provider can forward them, after checking the warrant. One can try to avoid surveillance by using expiring messages, but service providers can be forced to keep encrypted messages for a certain period of time. Or law enforcement can actively record messages in transit.

Our construction makes use of an extractable witness encryption scheme Π_{EWE} to encrypt the law enforcement ciphertext c_2. This scheme is parameterized by a language L_{EWE} that is defined with respect to the transparency function $t(\cdot)$, the policy function $p(\cdot)$, the targeting function $\theta(\cdot,\cdot)$, the warrant signing key $\mathsf{pk}_{\text{sign}}$, and the ledger verification function $\mathcal{L}.\mathsf{Verify}$, as follows:

$$L_{\text{EWE}} = \left\{ \mathsf{meta} \;\middle|\; \exists w, (t, \mathsf{info}, \pi_{\mathsf{publish}}) \text{ s.t. } \begin{array}{r} w = (\hat{w}, \sigma), \mathcal{L}.\mathsf{Verify}((\ell \| \mathsf{info}), \pi_{\mathsf{publish}}) = 1, \\ \mathsf{info} = t(w), \Pi_{\text{Sign}}.\mathsf{Verify}(\mathsf{pk}_{\text{sign}}, \hat{w}, \sigma) = 1, \\ p(\hat{w}) = 1, \theta(\hat{w}, \mathsf{meta}) = 1 \end{array} \right\}$$

Intuitively, these ciphertexts can only be decrypted by law enforcement once they have performed all the accountability tasks required by the ARLEAS.

We will describe our protocol in a hybrid model that makes use of several functionalities. These include $\mathcal{L}$, $\mathcal{F}_{CRS}^{\mathcal{D}}$, $\mathcal{G}_{\mathsf{pRO}}$ and $\mathcal{F}_{AUTH}$.

6.1 UC-Realizing $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{ret}}$

We give a description of the retrospective ARLEAS protocol $\pi_{\text{RET}}^{v,t,p,\theta}$ in Fig. 5.

Theorem 2. *Assuming a CCA-secure public key encryption scheme Π_{Enc}, an extractable witness encryption scheme for L_{EWE}, a SUF-CMA secure signature scheme Π_{Sign}, and a simulation-extractable NIZK scheme Π_{NIZK}, $\pi_{RET}^{v,t,p,\theta}$ (presented in Fig. 5) UC-realizes $\mathcal{F}_{ARLEAS}^{v,t,p,\theta,\mathsf{ret}}$ in the $\mathcal{L}^{\mathsf{Verify}}, \mathcal{F}_{CRS}^{\Pi_{NIZK}.\mathsf{ZKSetup}}, \mathcal{G}_{\mathsf{pRO}}$−hybrid model.*

Security Proof. We show the full security proof in the full version of the paper. The proof proceed similarly to the prospective case, with the exception that the simulator needs to equivocate on the context of ciphertexts once law enforcement is able to decrypt them. This equivocation is facilitated by the random oracle.

7 On the Need for Extractable Witness Encryption

The retrospective solution we present in Sect. 6 relies on extractable witness encryption. Intuitively, this strong assumption is required in our construction because a user must encrypt in a way that decryption is only possible under certain circumstances. Because the description of these circumstances can be phrased as an NP relation, witness encryption represents a "natural" primitive for realizing it. However, thus far we have not shown that the use of extractable witness encryption is *strictly* necessary. Given the strength (and implausibility [38]) of the primitive, it is important to justify its use. We do this by showing that any protocol Π_{A} that UC-realizes $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{ret}}$ implies the existence of extractable witness encryption for a related language. Notice that this does not mean the existence of a particular ARLEAS instantiation implies the existence of generic extractable witness encryption scheme, but rather a specific, non-trivial scheme.

Before proceeding to formally define this related language, we give some intuition about its form. We wish to argue that a protocol Π_{A} *acts like* an extractable witness encryption scheme in the specific case where an adversary has corrupted the escrow authorities P_{LE} and P_J (along with an arbitrary number of unrelated users). Recall that in order to learn any information about a message sent in Π_{A}, the following conditions must be met: specifically, law enforcement must correctly run the protocol for $\Pi_{\mathrm{A}}.\mathsf{RequestWarrant}$ and $\Pi_{\mathrm{A}}.\mathsf{ActivateWarrant}$ such that if $\Pi_{\mathrm{A}}.\mathsf{VerifyWarrantStatus}$ were to be called, it would output 1.[7] For the protocol we presented in Sect. 6, this corresponds to obtaining a correct proof of publication from the ledger. Importantly, it must be impossible for law enforcement and judges to generate this information independently; if it were possible, it would be easy for these parties to circumvent the accountability mechanism.

We give a formal definition of this language L below. We denote the view of a user P_i as $\mathcal{V}_{P_i}$, where this view is a collection of the views of running all algorithms that appear. We abuse notation slightly and denote the protocol transcript resulting from a sender P_S sending a message m to P_R as $\Pi_{\mathrm{A}}.\mathsf{SendMessage}(\cdot, P_S, P_R, m)$

$$L = \left\{ (\mathsf{meta}, \mathsf{sid}) \;\middle|\; \exists \left(w, c, \left\{ \begin{array}{c} \mathcal{V}_{P_{\mathrm{LE}}}, \mathcal{V}_{P_J}, \\ \{\mathcal{V}_{P_i}\}_{P_i \in \{P_1, \ldots, P_n\}/\{P_S, P_R\}} \end{array} \right\} \right) \text{ s.t.} \right.$$
$$\left. \begin{array}{r} c, \mathsf{meta} \leftarrow \Pi_{\mathrm{A}}.\mathsf{SendMessage}(\mathsf{sid}, P_S, P_R, m), \\ (\mathsf{Approve}) \leftarrow \Pi_{\mathrm{A}}.\mathsf{RequestWarrant}(\mathsf{sid}, w), \\ (\mathsf{NotifyWarrant}, t(w)) \leftarrow \Pi_{\mathrm{A}}.\mathsf{ActivateWarrant}(\mathsf{sid}, w), \\ 1 \leftarrow \Pi_{\mathrm{A}}.\mathsf{VerifyWarrantStatus}(\mathsf{sid}, w, \mathsf{meta}, c) \end{array} \right\}$$

In this language, the statement comprises some specified metadata and a valid instance of the protocol Π_{A} from the perspectives of the parties P_{LE}, P_J, and the users P_i without the sender and receiver. This setup specifies all the relevant components of the protocol (including the ledger functionality, in the case of the protocol presented in Sect. 6). The witness is a valid transcript starting with that setup, that includes the sending party sending a message with the appropriate metadata and concludes with a call to $\Pi_{\mathrm{A}}.\mathsf{VerifyWarrantStatus}$ that returns 1. Note that if $\mathsf{VerifyWarrantStatus}$ returns 1, then in the real protocol, $\mathsf{AccessMessage}$ would return the relevant plaintext. Unlike other common witness encryption languages, we note that all correctly sampled statements are trivially in the language and have multiple witnesses. Therefore, we need the strong notion of extractable witness encryption. As we will discuss, finding a witness for the statement remains a difficult task.

Consider the implications if it were computationally feasible for an adversary to generate a witness for an honestly sampled statement for L. This would imply that an adversary corrupting P_{LE} and P_J interacting with the real protocol has a correct witness, which includes a call to $\mathsf{ActivateWarrant}$, this implies

[7] As specified in the ideal functionality, during verification it will be checked that a warrant was properly requested and activated.

our accountability property. Such a protocol could never succeed in meeting our original goals; law enforcement would always be able to simulate the steps required for proper accountability. An accountability mechanism that can be locally simulated cannot guarantee that all parties can monitor the mechanism, undermining the purpose of the protocol.

To formalize this intuition, we begin by describing an extractable witness encryption scheme Π_{EWE} for language L given access to an ARLEAS protocol Π_{A}.

- $\mathsf{Enc}(x, m)$ parses $(\mathsf{meta}, \mathsf{sid})$ from x and calls $\Pi_{\text{A}}.\mathsf{SendMessage}(\mathsf{sid}, m, P_S, P_R)$ such that it outputs meta, c. It then returns the views $\{\mathcal{V}_{P_{\text{LE}}}, \mathcal{V}_{P_J}, \mathcal{V}_{P_0}, \ldots, \mathcal{V}_{P_n}\}$ resulting from that run, excluding the private information associated with sending the message.
- $\mathsf{Dec}(c, \omega)$ first parses $c, w, \mathsf{meta}, \mathsf{sid}$ from the inputs c and ω, then calls $m \leftarrow \Pi_{\text{A}}.\mathsf{AccessMessage}(\mathsf{sid}, w, \mathsf{meta}, c)$ and returns the result.

It is easy to see that this construction satisfies the correctness property of extractable witness encryption. Notice that a valid witness needs to contain inputs to VerifyWarrantStatus such that it outputs 1. Because VerifyWarrantStatus is defined to return 1 exactly when AccessMessage will return a message, the above decryption algorithm will return a message only with a valid witness.

We introduce the metadata in the statement in order to fix a witness to a particular statement. Note that our protocol generates an encryption as running part of the protocol, actually generating part of the witness. If metadata is not included in the statement, then *any* witness for a particular setup can be used to decrypt *any* ciphertext generated by the encryption oracle under the same statement. While this is not inherently problematic for extractable witness encryption, it no longer corresponds neatly to ARLEAS. Recall that warrants in ARLEAS specify the metadata for which they are relevant through the warrant scope check functionality $\theta(\cdot, \cdot)$ and this property must be enforced in the language. We now proceed to show that the above scheme Π_{EWE} satisfies extractable security if Π_{A} UC-realizes $\mathcal{F}_{\text{ARLEAS}}^{v,t,p,\theta,\mathsf{ret}}$.

Theorem 3. *Given a protocol Π_A that UC-realizes $\mathcal{F}_{ARLEAS}^{v,t,p,\theta,\mathsf{ret}}$, Π_{EWE} is a secure extractable witness encryption scheme for the language L.*

Proof. Given an adversary $\mathcal{A}$ with non-negligible advantage in the extractable witness encryption game for language L, either

1. We construct an extractor $\mathsf{Ext}_{\mathcal{A}}(1^{\lambda}, x, aux)$ by verifying if the adversary $\mathcal{A}$ ran $\Pi_{\text{A}}.\mathsf{RequestWarrant}(\mathsf{sid}, w)$ and $\Pi_{\text{A}}.\mathsf{ActivateWarrant}(\mathsf{sid}, w)$ such that $\Pi_{\text{A}}.\mathsf{VerifyWarrantStatus}(\mathsf{sid}, w, \mathsf{meta}, c) = 1$. If this was the case, the extractor would have all information to form a witness that it can output;
2. else, if such extractor does not exist, we construct a distinguisher $\mathcal{Z}$ that distinguishes between Π_{A} and ARLEAS ideal functionality. $\mathcal{Z}$ proceeds as follows

(a) When $\mathcal{A}$ asks to sample a statement, $\mathcal{Z}$ instantiates Π_{A} with parties $\{P_{\mathrm{LE}}, P_J, P_0, \ldots P_n, P_S, P_R\}$ on honest random coins. $\mathcal{Z}$ then generates some arbitrary metadata meta associated with a message that P_S could send in the future. and returns $\mathsf{meta}, \mathsf{sid}$ to $\mathcal{A}$.

(b) When $\mathcal{A}$ sends the challenge plaintexts m_0, m_1 (such that $|m_0| = |m_1|$) on statement x, $\mathcal{Z}$ then flips a coin $b \xleftarrow{\$} \{0,1\}$, $\mathcal{Z}$ has P_S call

$$\Pi_{\mathrm{A}}.\mathsf{SendMessage}(\mathsf{sid}, m_b, P_S, P_R)$$

such that it outputs c, meta. $\mathcal{Z}$ then returns the updated views of P_{LE}, P_J and the N other users to $\mathcal{A}$.

(c) When $\mathcal{A}$ outputs the guess b' and halts, $\mathcal{Z}$ outputs $b' == b$, where 1 indicates the real world and 0 indicates the ideal world.

Note that in the ideal functionality, the joint views of law enforcement and the judge contain no information about the plaintext, because the ciphertext is chosen by the ideal world adversary without access to the plaintext. As such, if the adversary is able to distinguish between messages with non-negligible probability, $\mathcal{Z}$ must be interacting with the real world protocol.

Implications For Practical Retrospective ARLEAS. The relationship between retrospective ARLEAS and extractable witness encryption is an indication of the difficulty of realizing retrospective ARLEAS in practice. In very specific cases, it may be possible to phrase certain existing encryption schemes as witness encryption schemes, for example some IBE schemes. General purpose extractable witness encryption, on the other hand, is considered implausible [38]. The extractable witness encryption language we have described above must reason over the ledger authentication language and the various functionalities that parameterize an retrospective ARLEAS system. As such, the difficulty of realizing a practical retrospective ARLEAS will hinge on the complexity of the ledger and the parameterizing functionalities. If they are centralized and simple, it may be possible to instantiate an retrospective ARLEAS using the protocol we provided in Sect. 6 and known encryption techniques. However, the security provided by a centralized ledger is not significant, as a compromised central authority could circumvent the accountability properties of the system. Thus, we believe that this result indicates that instantiating an retrospective ARLEAS with meaningful security is impractical with known techniques.

Acknowledgments. The first author funded in part from the National Science Foundation under awards CNS-1653110 and CNS-1801479, a Google Security & Privacy Award. The second author is supported by the National Science Foundation under Grant #2030859 to the Computing Research Association for the CIFellows Project. Additionally, this material is based upon work supported by DARPA under Agreements No. HR00112020021 and Agreements No. HR001120C0084. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

References

1. Abelson, H., et al.: Keys under doormats: mandating insecurity by requiring government access to all data and communications. J. Cybersecur. **1**(1), 69–79 (2015)
2. Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 387–404. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_22
3. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 443–458. IEEE Computer Society Press, May 2014
4. Apple. Facetime. https://apps.apple.com/us/app/facetime/id1110145091
5. Apple. icloud security overview. https://support.apple.com/en-us/HT202303
6. Apple. imessage. https://support.apple.com/explore/messages
7. Backes, M., Camenisch, J., Sommer, D.: Anonymous yet accountable access control. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, pp. 40–46. Association for Computing Machinery, New York (2005)
8. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 913–930. ACM Press (2018)
9. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: a composable treatment. In: Katz, J., Shacham, H. (eds.) CRYPTO, Part I. LNCS, vol. 10401, pp. 324–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_11
10. Barr, W.: Attorney general William P. Barr delivers keynote address at the international conference on cyber security, July 2019
11. Barr, W.: Attorney general William P. Barr delivers keynote address at the international conference on cyber security, July 2019. https://www.justice.gov/opa/speech/attorney-general-william-p-barr-delivers-keynote-address-international-conference-cyber
12. Bates, A.M., Butler, K.R.B., Sherr, M., Shields, C., Traynor, P., Wallach, D.S.: Accountable wiretapping -or- I know they can hear you now. In: NDSS 2012. The Internet Society, February 2012
13. Bellare, M., Rivest, R.L.: Translucent cryptography - an alternative to key escrow, and its implementation via fractional oblivious transfer. J. Cryptol. **12**(2), 117–139 (1999)
14. Bellovin, S.M., Blaze, M., Boneh, D., Landau, S., Rivest, R.R.: Analysis of the CLEAR protocol per the National Academies' framework. Technical report CUCS-003-18, Columbia University, May 2018
15. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 421–439. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_24
16. Blaze, M.: Oblivious key escrow. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 335–343. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61996-8_50
17. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25

18. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_3
19. Bryan-Low, C.: Vodafone, Ericsson get hung up in Greece's phone-tap scandal. Wall Street J. (2006)
20. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 280–312. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_11
21. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
22. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33
23. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press, May 2002
24. Choudhuri, A.R., Goyal, V., Jain, A.: Founding secure computation on blockchains. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 351–380. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_13
25. Choudhuri, A.R., Green, M., Jain, A., Kaptchuk, G., Miers, I.: Fairness in an unfair world: fair multiparty computation from public bulletin boards. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 719–728. ACM Press, October/November 2017
26. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
27. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially optimal universally composable oblivious transfer. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 318–335. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00730-9_20
28. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 66–98. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_3
29. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_33
30. Denning, D.E.: The US key escrow encryption technology. Comput. Commun. **17**(7), 453–457 (1994)
31. Denning, D.E., Branstad, D.K.: A taxonomy for key escrow encryption systems. Commun. ACM **39**(3), 34–40 (1996)
32. EncroChat. Encrochat network. http://encrochat.network/
33. Encryption Working Group: Moving the Encryption Policy Conversation Forward. Technical report, Carnegie Endowment for International Peace (2019)
34. Federal Bureau of Investigation. Going Dark. https://www.fbi.gov/services/operational-technology/going-dark

35. Feigenbaum, J., Weitzner, D.J.: On the incommensurability of laws and technical mechanisms: or, what cryptography can't do. In: Matyáš, V., Švenda, P., Stajano, F., Christianson, B., Anderson, J. (eds.) Security Protocols 2018. LNCS, vol. 11286, pp. 266–279. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03251-7_31
36. Franceschi-Bicchierai, L.: FBI director: encryption will lead to a 'very dark place'. Mashable, October 2014
37. Frankle, J., Park, S., Shaar, D., Goldwasser, S., Weitzner, D.J.: Practical accountability of secret processes. In: Enck, W., Felt, A.P. (eds.) USENIX Security 2018, pp. 657–674. USENIX Association, August 2018
38. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 518–535. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_29
39. Garg, S., Ostrovsky, R., Visconti, I., Wadia, A.: Resettable statistical zero knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 494–511. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_28
40. Gazi, P., Kiayias, A., Zindros, D.: Proof-of-stake sidechains. In: 2019 IEEE Symposium on Security and Privacy, pp. 139–156. IEEE Computer Society Press, May 2019
41. Gentry, C., Lewko, A., Waters, B.: Witness encryption from instance independent assumptions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 426–443. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_24
42. Goldwasser, S., Park, S.: Public accountability vs. secret laws: can they coexist? A cryptographic proposal. In: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, WPES 2017, pp. 99–110. Association for Computing Machinery, New York (2017)
43. Google. Encrypt your data - pixel phone help. https://support.google.com/pixelphone/answer/2844831?hl=en
44. Gorman, S.: NSA officers spy on love interests. Wall Street J. (2013)
45. Goyal, R., Goyal, V.: Overcoming cryptographic impossibility results using blockchains. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 529–561. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_18
46. Graham, S.L.: Eliminating abusive and rampant neglect of interactive technologies act of 2020, March 2020
47. Horel, T., Park, S., Richelson, S., Vaikuntanathan, V.: How to subvert backdoored encryption: security against adversaries that decrypt all ciphertexts. In: Blum, A. (ed.) ITCS 2019, vol. 124, pp. 42:1–42:20. LIPIcs (2019)
48. Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_7
49. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_23
50. Kamara, S.: Restructuring the NSA metadata program. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 235–247. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_19

51. Kaptchuk, G., Green, M., Miers, I.: Giving state to the stateless: augmenting trustworthy computation with ledgers. In: NDSS 2019. The Internet Society, February 2019
52. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
53. Kroll, J., Felten, E., Boneh, D.: Secure protocols for accountable warrant execution (2014)
54. Kroll, J.A., Zimmerman, J., Wu, D.J., Nikolaenko, V., Felten, E.W., Boneh, D.: Accountable cryptographic access control (2018)
55. Levy, I., Robinson, C.: Principles for a more informed exceptional access debate. Lawfare (2018)
56. Lichtblau, E., Goldstein, J.: Apple faces U.S. demand to unlock 9 more iPhones. The New York Times, February 2016
57. Liu, J., Ryan, M.D., Chen, L.: Balancing societal security and individual privacy: accountable escrow system. In: 2014 IEEE 27th Computer Security Foundations Symposium, pp. 427–440, July 2014
58. Liu, J., Jager, T., Kakvi, S.A., Warinschi, B.: How to build time-lock encryption. Des. Codes Crypt. **86**(11), 2549–2586 (2018)
59. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
60. Nakashima, E.: Chinese hackers who hacked Google gained access to sensitive data, U.S. officials say. The Washington Post, May 2013
61. National Academies of Sciences, Engineering, and Medicine. Exploring Encryption and Potential Mechanisms for Authorized Government Access to Plaintext, The National Academies Press (2016)
62. National Academies of Sciences, Engineering, and Medicine. Decrypting the Encryption Debate: A Framework for Decision Makers: The National Academies Press, Washington, DC (2018)
63. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_22
64. Nightingale, J.: Fraudulent *.google.com Certificate, August 2011
65. Panwar, G., Vishwanathan, R., Misra, S., Bos, A.: SAMPL: scalable auditability of monitoring processes using public ledgers. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019, pp. 2249–2266. ACM Press, November 2019
66. Poplin, C.M.: Burr-feinstein encryption legislation officially released. Lawfare, April 2016
67. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, pp. 543–553. IEEE Computer Society Press, October 1999
68. Savage, S.: Lawful device access without mass surveillance risk: a technical design discussion. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp. 1761–1774. Association for Computing Machinery, New York (2018)
69. Scafuro, A.: Break-glass encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 34–62. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_2

70. Segal, A., Ford, B., Feigenbaum, J.: Catching bandits and only bandits: Privacy-preserving intersection warrants for lawful surveillance. In: 4th USENIX Workshop on Free and Open Communications on the Internet (FOCI 14). USENIX Association, San Diego, CA, August 2014
71. Blackburn, Sen.M., Graham, Sen.L., Cotton, Sen.T.: Lawful access to 5 encrypted data act, June 2020
72. Servan-Schreiber, S., Wheeler, A.: Judge, jury & encryptioner: exceptional access with a fixed social cost (2019)
73. Signal. Signal secure messaging system
74. Sing, M.: Over two dozen encryption experts call on India to rethink changes to its intermediary liability rules. TechCrunch, February 2020
75. Tait, M.: An approach to James Comey's technical challenge. Lawfare, April 2016
76. Tarabay, J.: Australian government passes contentious encryption law. The New York Times, December 2018
77. Watt, N., Mason, R., Traynor, I.: David Cameron pledges anti-terror law for internet after Paris attacks. The Guardian, January 2015
78. WhatsApp. WhatsApp Encryption Overview, December 2017
79. Wright, C., Varia, M.: Crypto crumple zones: enabling limited access without mass surveillance. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P), pp. 288–306, April 2018