How to (Legally) Keep Secrets from Mobile Operators

Ghada Arfaoui[1], Olivier Blazy[2], Xavier Bultel[3], Pierre-Alain Fouque[4],

Thibaut Jacques[4], Adina Nedelcu[1,4], and Cristina Onete[2(✉)]

[1] Orange Labs, Cesson-Sévigné, France
[2] XLIM, University of Limoges, Limoges, France
[3] INSA Centre Val-de-Loire, Bourges, France
[4] IRISA, University of Rennes 1, Rennes, France

Abstract. Secure-channel establishment allows two endpoints to communicate confidentially and authentically. Since they hide all data sent across them, good or bad, secure channels are often subject to mass surveillance in the name of (inter)national security. Some protocols are constructed to allow easy data interception . Others are designed to preserve data privacy and are either subverted or prohibited to use without trapdoors.

We introduce LIKE, a primitive that provides secure-channel establishment with an exceptional, session-specific opening mechanism. Designed for mobile communications, where an operator forwards messages between the endpoints, it can also be used in other settings. LIKE allows Alice and Bob to establish a secure channel with respect to n authorities. If the authorities all agree on the need for interception, they can ensure that the session key is retrieved. As long as at least one honest authority prohibits interception, the key remains secure; moreover LIKE is versatile with respect to who learns the key. Furthermore, we guarantee non-frameability: nobody can falsely incriminate a user of taking part in a conversation; and honest-operator: if the operator accepts a transcript as valid, then the key retrieved by the authorities is the key that Alice and Bob should compute. Experimental results show that our protocol can be efficiently implemented.

1 Introduction

For almost a decade mass surveillance has caused controversy, widespread protests, and scandals; and yet, it is on the rise. The NSA, for instance, illegally collected phone-call data from all Verizon customers, and the data of all calls occurring in the Bahamas and Afghanistan [28]. During the present COVID-19 pandemic, Germany used contact-tracing data to pursue criminal investigations [26]. The need for privacy-enhancing solutions that provide transparency and limit mass surveillance has never been greater.

User privacy is a human right acknowledged by Article 12 of the Universal Declaration for Human Rights [42]: "No one shall be subjected to arbitrary interference with his privacy [...] or correspondence [...]. Everyone has the right to the protection of the law against such [...] attacks." The European General Data Protection Regulation (GDPR) and e-Privacy both aim to protect privacy in digital environments, requiring

minimal, transparent, secure, and user-controlled storage of data. Increasingly aware of mass surveillance, users now choose more frequently to secure their communications by using, *e.g.*, WhatsApp and Viber. In mobile networks, data is encrypted by mobile network operators, which have an incentive to improve the privacy they offer their users.

Unfortunately, mobile data remains at risk, especially when exceptional access to it lies with a single entity. In 2016, only the integrity of Tim Cook (Apple CEO) and his awareness of the danger of such a precedent prevented encrypted phone data to be given to the FBI [30]. His refusal was not a deterrent, and risks to privacy and encryption grow every day [25].

Law-enforcement agencies argue that mobile data can be pivotal to investigations, which are undermined by individuals "going dark" [21]. Even strong privacy advocates, such as Abelson *et al.* [6] agree that *targeted* investigations, *limited in scope and motivation*, can be legitimate and useful (see the corruption-scandal regarding Nicolas Sarkozy's campaign funds). It is this type of limited Lawful Interception (LI) that emerging EU legislation advocates [24].

Recent research [45] has tried to find a middle ground between privacy and lawful interception, enabling the latter at high cost. Unfortunately, that approach sacrifices a crucial real-world requirement: timely (exceptional) decryption [2]. In this paper we define and instantiate Lawful-Interception Key-Exchange (LIKE), a novel primitive allowing mobile users to E2E encrypt their conversation, but providing exceptional lawful interception. This would remove the "[...] need to choose between compliance and strong encryption" (cf. Joel Wallenstrom).

LIKE combats mass surveillance in multiple ways. It *fine-grains* the window of interception to a single session, such that one exceptional opening will not affect past or future conversations. Moreover, the freshness used in the protocol is user- not authority-generated, thus *removing the need* for a centralized, secure party storing all the keys. We also divide the responsibility of exceptional opening between multiple authorities that *must agree* to lawfully intercept communications. Finally, we make the primitive *versatile*, allowing only one, or only several authorities to ultimately retrieve the session key. This makes our solution more privacy-preserving than mobile protocols today, while at the same time remaining compatible to the strictest LI-supporting 3GPP texts [2].

Lawful Interception. Regardless of one's stand on it, Lawful Interception (LI) is part of our world, regulated by laws and standards. Ignoring it can lead to privacy threats (subversion or mass surveillance). Solutions that provide privacy but no LI are discarded as unlawful, regardless of their merits. We take the alternative approach: we analyze LI requirements and technically provide for them, while still maximizing data privacy.

LI requirements in mobile communications are authored by 3GPP and standardized by organizations, *e.g.*, ETSI. LI is the procedure through which a law enforcement agency, holding a legal warrant, can obtain information about phone calls: either metadata (time of calls, identity of callers) or contents (of conversations happening in real time). By law, a mobile operator *must* provide the data requested and specified by a warrant. Three main types of requirements regulate LI: user-privacy requirements, LI-security requirements, and requirements on encryption.

R1 User Privacy: the interception is limited in time (as dictated by the warrant) and to a targeted user. The law enforcement agency should not get data packages or conversations outside the warrant, from the same or other users.

R2 LI Security: LI must be undetectable by either users (whose quality of service should stay the same) or non-authorized third parties (including other law-enforcement agencies). Intercepted data must be provided promptly, with no undue delay.

R3 Special case: if it implements encryption, the operator must provide either decrypted content, or a means to decrypt it (*e.g.*, a decryption key). However, if the users employ other means of encryption, not provided by the operator, the latter is not obliged to provide decrypted (or decryptable) conversations.

1.1   Our Contributions

LIKE. Based on these requirements, we define a novel cryptographic primitive called Lawful-Interception Key Exchange (LIKE, in short). This protocol allows (only) the end users to compute session keys in the presence of a variable number n of authorities (*e.g.*, a court of justice, a law enforcement agency, operators), which all parties must agree on; the operators output a public *session state*. Given the session state, authorities may each extract a trapdoor. The use of all n trapdoors can yield a session key. Importantly, unless they are an authority, operators *cannot* recover the end users' key; instead they forward and verify the compliance of exchanged messages (else the operator aborts).

We also formalize the following strong properties:

C Correctness: Under normal conditions, Alice and Bob obtain the same key. Moreover, this will be the key retrieved by the collaboration of all the authorities by means of lawful interception (requirement R2)

KS Key-security: If at least one authority and both users are honest for a given session, that session's key remains indistinguishable from a random key of the same length with respect to an adversary that can control all the remaining parties (including the other authorities and the operator); (requirements R1 and R2)

NF Non-frameability: The collusion of malicious users, the authorities, and the operator cannot frame an honest user of participating to a session she has not been a party to (requirement R2);

HO Honest Operator: If an honest operator forwards the so-called session state (see below) of a session it deems correct, then the key recovered by the authorities is the one that the session transcript should have yielded (requirement R3). Thus operators can prove that this protocol is compliant with LI specifications.

Our Protocol and Implementation. As our second contribution, we describe an instantiation of LIKE using standard building blocks (signatures, zero knowledge proofs and signatures of knowledge) which we prove secure, provided the Bilinear Decisional Diffie-Hellman problem is intractable, the signature scheme is unforgeable, and our zero-knowledge proofs/signatures are secure.

Mindful of practical requirements, we place most of the burden during AKE on the operator (not on the endpoints). Our proofs and signatures of knowledge can be simply implemented based on Schnorr and respectively Chaum and Pedersen proofs (with Fiat-Shamir). The two endpoints do have to compute a pairing operation –however, we explain that the actual computation can actually be delegated to the mobile phone, leaving a single exponentiation (in the target group) to be performed on the USIM card.

The complexity of the opening procedure is reduced. Some steps are parallelizable and run in constant time (trapdoor generation), but others are linear (combining the trapdoors). Even so the computational burden remains minimal and in line with requirements R2 (constant quality of service, since only authorities run LI, not the operator) and R3 (no undue delay would be incurred by the operator, and only minimal delays occur at the authorities). A proof-of-concept implementation given in Sect. 7 illustrates this point.

1.2   Related Work

Existing encryption in mobile networks is not E2E secure, only providing privacy with respect to non-authorized third parties. This solution is compliant to LI requirements [3–5] because the secure channel it provides is between the user and the operator (rather than user-to-user); thus the operator has unrestricted access to all user communications. Our LIKE protocol provides much stronger privacy in that respect.

LIKE also provides much stronger guarantees than key-escrow [8,11,19,20,22,27, 32–35,37,38,40,41,44]: we can handle malicious authority input; we fine-grain exceptional opening so that it only holds for one session at a time; we allow authorities to remain offline at all times except for exceptional opening; we minimize storage and computational costs for users and authorities; we have no central key-generation authority which knows all secret keys; and we guarantee the new properties of non-frameability and honest operator, which are tailored to the LI requirements analyzed above. In return, our use-case is narrower than typically considered in key-escrow: by considering the case of mobile communications, we can safely assume that parties always use the operator as a proxy, unlike in generic key-escrow settings.

Our work is motivated by the same problem handled by [12,45], but we take a different approach: rather than make LI computationally costly, we limit its scope, divide exceptional opening between several authorities, and fine-grain access. Our setup also resembles that of reverse firewalls [36] – which builds on related work pioneered by Young and Yung [23,31,46]. Although apparently similar, the setup of these works is complementary to ours. Kleptography describes ways for users to abuse protocols such that the latter appear to be running normally, while subliminal information is allowed to leak. For instance in key-exchange, a government agency could substitute the implementation of the protocol by one with a backdoor, to make the key recoverable even without formal LI. We also consider the case that Alice and Bob might be malicious (the HO property). However, rather than wanting to exfiltrate information –which is the adversarial goal in typical reverse-firewalls– in our case, Alice and Bob aim to cheat by choosing protocol contributions that might make LI malfunction. To be provably-secure, LIKE protocols must prevent this.

2 Preliminaries

Notations. By $x \leftarrow y$ we mean that variable x takes a value y, while $x \xleftarrow{\$} X$ indicates x is chosen from the uniform distribution on X. The notation $[\![1, n]\!]$ is short for $\{1, 2, \cdots, n\}$. Let $\mathsf{A}(x) \to a$ express that algorithm A, running on input x, outputs a, and $\mathsf{P}\langle \mathsf{A}(x), \mathsf{B}(y)\rangle(z) \to (a, b)$ to express that protocol P implements the interactions of $\mathsf{A}(x) \to a$ and $\mathsf{B}(y) \to b$, where z is an additional public input of A and B. Let λ be a security parameter.

Building Blocks. We assume the reader's familiarity with the notion of unforgeability for signature schemes (EUF-CMA), also visible in our full version [7]. We denote by $\mathsf{Adv}_{\mathsf{DS}}^{\mathsf{EUF\text{-}CMA}}(\lambda)$ the maximal advantage an adversary has against the signature scheme that we employ.

Definition 1 (BDDH Assumption [15]**).** *Let* $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, *and* $\mathbb{G}_T$ *be groups of prime order* p *of length* λ. *Let* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *be a type 3 bilinear map. The* Bilinear decisional Diffie-Hellman problem *(BDDH) assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *if, given* $(a, b, c, d_1) \xleftarrow{\$} (\mathbb{Z}_p^*)^4$, $d_0 \leftarrow abc$, *and* $\beta \xleftarrow{\$} \{0, 1\}$, *no PPT adversary* $\mathcal{A}$ *can guess* β *from* $(g_1^a, g_2^a, g_1^b, g_2^b, g_1^c, g_2^c, e(g_1, g_2)^{d_\beta})$ *with non-negligible advantage. We denote by* $\mathsf{Adv}^{\mathsf{BDDH}}(\lambda)$ *the maximum advantage over all PPT adversaries.*

Signature of Knowledge. Let $\mathcal{R}$ be a binary relation and let $\mathcal{L}$ be a language such that $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. A Non-Interactive Proof of Knowledge (NIPoK) [10] allows a prover to convince a verifier that he knows a witness w such that $(s, w) \in \mathcal{R}$. Here, we follow [16] and write $\mathsf{NIPoK}\{w : (w, s) \in \mathcal{R}\}$ for the proof of knowledge of w for the statement s and the relation $\mathcal{R}$. A *signature of knowledge* essentially allows one to sign a message and prove in zero-knowledge that a particular statement holds for the key [17]. In this paradigm, w is a secret key and s is the corresponding public key.

Definition 2 (Signature of Knowledge). *Let* $\mathcal{R}$ *be a binary relation and* $\mathcal{L}$ *be a language such that* $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. *A* Signature of Knowledge *for* $\mathcal{L}$ *is a pair of algorithms* $(\mathsf{SoK}, \mathsf{SoKver})$ *with* $\mathsf{SoK}_m \{w : (s, w) \in \mathcal{R}\} \to \pi$ *and* $\mathsf{SoKver}(m, s, \pi) \to b$, *such that:*

- *Perfect Zero Knowledge: There exists a polynomial time algorithm* **Sim**, *the simulator, such that* **Sim**(m, s) *and* $\mathsf{SoK}_m \{w : (s, w) \in \mathcal{R}\}$ *follow the same probability distribution.*
- *Knowledge Extractor: There exists a PPT knowledge extractor* **Ext** *and a negligible function* ϵ_{SoK} *such that for any algorithm* $\mathcal{A}^{Sim(\cdot,\cdot)}(\lambda)$ *having access to a simulator that forges signatures for chosen instance/message tuples and that outputs a fresh tuple* (s, π, m) *with* $\mathsf{SoKver}(m, s, \pi) = 1$, *the extractor* **Ext**$^A(\lambda)$ *outputs* w *such that* $(s, w) \in \mathcal{R}$ *having access to* $\mathcal{A}(\lambda)$ *with probability at least* $1 - \epsilon_{\mathsf{SoK}}(\lambda)$.

We omit the definition of NIPoK which is the same as SoK without the messages.

3   LIKE Protocols

Lawful-interception (authenticated) key-exchange (LIKE) consists of two mechanisms: a multiparty AKE protocol between 2 mobile subscribers (users) and a (number of) mobile network operators (performing the same actions), and an Extract-and-Open mechanism for LI between a number of *authorities*.

Intuition. Our AKE component allows user Alice, subscribing to operator O_A, and Bob, subscribing to O_B, to compute a session key in the presence of O_A and O_B. The operators do not compute the session key, just some auxiliary *session state*, which allows for session authentication and ulterior key-recovery.

In the Extract-and-Open component, each authority uses its secret key to *extract* a trapdoor from the session state. The trapdoors are then used together to *open* the session key.

Formalization. Let USERS be a set of mobile users and OPS be a set of operators, such that each user is affiliated to precisely one operator. We also consider a set of authorities AUTH of cardinality |AUTH|, with elements indexed as $\Lambda_1, \ldots, \Lambda_{|AUTH|}$.

Let PARTIES be the set of all participants: USERS $\cup$ OPS $\cup$ AUTH. Mobile users have no super-role: USERS $\cap$ AUTH $= \emptyset =$ USERS $\cap$ OPS. Syntactically, OPS $\cap$ AUTH $= \emptyset$; however, operators can act as authorities by registering a second set of (authority) credentials and using those for opening. This is described in the full version.

Parties, attributes, and oracles are formally introduced in the next section. We use a dot notation to refer to *attributes* of a party P: A.PKis Alice's public key and Λ_i.SK is the secret key of the i-th authority. We write O_A to indicate Alice's operator even though we have no user registration and can run the protocol with *any* two operators chosen by the users. The operators are assumed to transit all communication (as is the case today). The parties can agree on a variable number n of (distinct) authorities, with $1 \leqslant n \leqslant |AUTH|$.

Definition 3. *A* lawful interception key exchange (LIKE) *protocol is defined by the following algorithms:*

- Setup(1^λ) $\rightarrow$ pp*: Takes as input a security parameter and outputs public system parameters* pp*, known to all parties.*
- UKeyGen(pp) $\rightarrow$ (U.PK, U.SK)*: Takes as input the public parameters* pp *and outputs a user key pair.*
- OKeyGen(pp) $\rightarrow$ (O.PK, O.SK)*: Takes as input the public parameters* pp *and outputs an operator key pair.*
- AKeyGen(pp) $\rightarrow$ (Λ.PK, Λ.SK)*: Takes as input the public parameters* pp *and outputs an authority key pair.*
- AKE$\langle$A(A.SK), $O_A(O_A.SK)$, $O_B(O_B.SK)$, B(B.SK)$\rangle$(PK$_{A\rightarrow B}$) $\rightarrow$ (k_A, sst$_A$, sst$_B$, k_B)*: An authenticated key-exchange protocol between users* (A, B) $\in$ USERS^2 *and their operators* $(O_A, O_B) \in$ OPS^2*, the latter providing active middleware at all times. The parties each take as input a secret key, and they all have access to the same set of public values* PK$_{A\rightarrow B}$ *containing: parameters* pp*, public keys* (A.PK, B.PK)*, and a vector of authority public keys* APK $= (\Lambda_i.PK)_{i=1}^n$ *with (distinct)* $\Lambda_i \in$ AUTH *for all* i*. At the end of the protocol,* A *(resp.* B*) returns a* session

secret key k_A *(resp. k_B) and the operator O_A (resp. O_B) returns a (public) session state sst_A (resp. sst_B). In case of failure, the parties output a special symbol $\perp$ instead.*

- Verify$(pp, sst, A.PK, B.PK, O.PK, APK) \rightarrow b$: *Takes as input session state sst, user public keys $A.PK$ and $B.PK$, an operator public key $O.PK$, a set of authority public keys$APK = (\Lambda_i.PK)_{i=1}^n$, outputting a bit $b = 1$ if sst was correctly generated and authenticated by O, and $b = 0$ otherwise.*
- TDGen$(pp, \Lambda.SK, sst) \rightarrow \Lambda.t$: *Takes as input authority secret key $\Lambda.SK$ and session state sst, and outputs trapdoor $\Lambda.t$.*
- Open$(pp, sst, APK, \mathcal{T}) \rightarrow k$: *Takes as input session state sst, two vectors $APK = (\Lambda_i.PK)_{i=1}^n$ with n distinct public keys, and $\mathcal{T} = (\Lambda_i.t)_{i=1}^n$ (authority public keys and corresponding trapdoors), and outputs either a session key k, or a symbol $\perp$.*

We defer the full correctness definition to Appendix A. It essentially states that if parameters are normally generated, for an honestly-run protocol session: both operators approve it, both users accept computing the same key, and that same key is extracted from either of the operators' session states.

To use a LIKE scheme, keys are generated based on the parties' role: UKeyGen for users, OKeyGen for operators, and AKeyGen for authorities. Then, Alice, Bob, and the operators run AKE as described above. At the end, Alice and Bob share a session key (unknown to the operator), and each operator returns a public session state sst. The values included in sst are protocol-dependent, and they must allow the authorities to verify that the session was run correctly and recover the session key. The verification is not exclusively meant for authorities: the algorithm Verify checks the validity of the session and authenticates its participants.

The authorities may later retrieve session keys. Each authority verifies the soundness of a given sst, then extracts a trapdoor to the session key, using its secret key. Given all the trapdoors, the algorithm Open retrieves the session key. Depending on the LI scenario, the Open algorithm may be run by one or multiple parties (thus, whoever has all the trapdoors extracts the key). Our protocol is versatile and can adapt to many cases, some of which are shown in the full version.

4   Security Model

This section formalizes LIKE security: an essential contribution to the paper, which provides much stronger guarantees than regular authenticated key exchange or key-escrow. We begin by giving the adversarial model, then list the oracles which adversaries can use to manipulate honest parties, and formalize security games for each property.

The Adversarial Model. We assume all parties (users, operators, authorities) have unique and unchanging roles. Parties may still play multiple roles if the same physical entity registers as multiple users in our scheme (an authority and an operator, for instance). Each party P is associated with these attributes:

- (SK, PK): long-term private (resp. public) keys SK (resp. PK). Such keys are output by UKeyGen, OKeyGen, or AKeyGen.

- γ: a corruption flag, indicating whether that party has been corrupted (1) or not (0). The flag starts out as 0, and if it changes to 1, it can never flipped back to 0.

Alice, Bob, and their operators run AKE in *sessions*. At each new session, a new *instance* of each party is created (yielding four instances per session). We denote by π_P^i the i-th instance generated during the experiment, where P denotes the corresponding party. In addition to the long-term keys and corruption bits of the party, instances keep track of the following attributes:

- sid: a session identifier consisting of a tuple of session-specific values, like public parameters or randomness. This attribute stores only state pertinent to that session, does not include secret values, and is computable by the instances running the session. The value of sid is initially set to $\bot$ and changes during the protocol run.
- PID: partner identifiers. If $P \in USERS$, then $PID \in USERS$ such that $P \neq PID$, else $P \in OPS$ and $PID \in USERS^2$ including two distinct parties.
- OID: operator identifiers. If $P \in USERS$ then $OID \in OPS^2$. Else $OID \in OPS$.
- AID: distinct authority identifiers such that $AID \in AUTH^n$.
- α: an accept flag, undefined ($\alpha = \bot$) until the instance terminates, either in an abort (setting $\alpha = 0$) or without error ($\alpha = 1$).
- k: the session key, initialized to $\bot$, and modified if the protocol terminates without error. Operator instances do not have this attribute.
- sst: a set of values pertaining to the instance's view of the session, initialized to $\bot$ and modified if the protocol terminates without error. User instances do not have this attribute.
- ρ: a reveal bit, initialized to 0 and set to 1 if the adversary reveals a session key $k \neq \bot$. Operator instances do not have this attribute.
- b: a bit chosen uniformly at random upon the creation of the instance.
- τ: the transcript of the session, initialized as $\bot$, turning to the ordered list of messages sent and received by that instance in the same order.

An auxiliary function IdentifySession (sst, π) is defined, taking as input session state sst and a party instance π, outputting 1 if π took part in the session where sst was created, and 0 otherwise. We need this at opening, when authorities must extract and verify the session identifier for themselves.

Notice that, unlike in typical AKE, we have two types of parties running each session (users and operators), and three attributes that describe partners: mobile user partners in PID, operator partners in OID, and authorities partnering it in AID. Moreover, IdentifySession and sst are protocol dependent, *i.e.*, they will have a different instantiation depending on the protocol we analyse.

We define matching conversation defined as follows:

Definition 4 (Matching Instances). *For any $(i, j) \in \mathbb{N}^2$ and $(A, B) \in USERS^2$ such that $A \neq B$, we say that π_A^i and π_B^j have matching conversation if all the following conditions hold: $\pi_A^i.sid \neq \bot$, $\pi_A^i.sid = \pi_B^j.sid$, and $\pi_A^i.AID = \pi_B^j.AID$. If two instances π_A^i and π_B^j have matching conversation, we sometimes say, by abuse of language, that π_A^i matches π_B^j.*

Oracles. We define LIKE security in terms of games played by an adversary plays against a challenger. In each game, the adversary $\mathcal{A}$ may query some or all of the oracles below. Intuitively, $\mathcal{A}$ may register honest or malicious participants (using Register), initiate new sessions (using NewSession), interact in the AKE protocol (using Send), corrupt parties (using Corrupt), reveal session keys (using Reveal), or reveal LI trapdoors (using RevealTD). Finally, the adversary has to query a testing oracle (Test) and tell whether the output was the real session key or a random one. Each oracle *aborts* if queried with ill-formatted input, or if insufficient information exists for the response.

- Register(P, `role`, PK) $\rightarrow \bot \cup$ P.PK: On input party P $\notin$ USERS $\cup$ OPS $\cup$ AUTH, role `role` $\in \{$`user`, `operator`, `authority`$\}$, and public key PK:
    - If `role` = `user` (resp. `operator`, `authority`), add P to the set USERS (resp. OPS, AUTH).
    - If `role` = `user` (resp. `operator` and `authority`) and PK = $\bot$, run UKeyGen(pp) $\rightarrow$ (P.PK, P.SK) (resp. OKeyGen and AKeyGen).
    - If PK $\neq \bot$, set the P.$\gamma = 1$, P.PK = PK, and P.SK = $\bot$.
  Finally, return P.PK.
- NewSession(P, PID, OID, AID) $\rightarrow \pi_P^i$: On input party P $\in$ USERS $\cup$ OPS, if P $\in$ USERS then PID, OID and AID must be such that: PID $\in$ USERS with P $\neq$ PID; OID $\in$ OPS^2; and AID $\in$ AUTH* with $|$AID$| \neq 0$. If P $\in$ OPS, then PID, OID, and AID are such that PID $\in$ USERS^2; OID $\in$ OPS; and AID $\in$ AUTH* such that $|$AID$| \neq 0$. On the i-th call to this oracle, return new instance π_P^i with already-set values for PID, OID, and AID.
- Send(π_P^i, m) $\rightarrow m'$: Send message m to instance π_P^i and return message m' according to protocol (potentially $\bot$ for inexistent, aborted, or terminated instance, for ill-formed m, or if P.SK = $\bot$).
- Reveal(π_P^i) $\rightarrow$ k: For accepting user instance π_P^i, return the session key π_P^i.k and set $\pi_P^i.\rho = 1$. For accepting operator instance π_P^i return the session state π_P^i.sst and set $\pi_P^i.\rho = 1$. If π_P^i is not an accepting instance ($\alpha \neq 1$), return $\bot$.
- Corrupt(P) $\rightarrow$ P.SK: Return P.SK of input party P (all roles) and set P.$\gamma = 1$ for all instances of this party.
- Test(π_P^i) $\rightarrow \tilde{\text{k}}$: Return $\bot$ if input instance π_P^i is not an accepting user instance or if this oracle has been queried before, outputting a value that is non-$\bot$. Else, if π_P^i.b = 0, return π_P^i.k, and otherwise return randomly-sampled r from the same domain as π_P^i.k. This oracle can only be called once during an experiment, which means that only one instance bit b is actually used.
- RevealTD(sst, A, B, O, $(\Lambda_i)_{i=1}^n$, l) $\rightarrow \Lambda_l.t$: If Verify(pp, sst, A.PK, B.PK, O.PK, $(\Lambda_i.$PK$)_{i=1}^n$) = 1, run $\Lambda_l.t \leftarrow$ TDGen(pp, Λ_l.SK, sst), return $\Lambda_l.t$; else return $\bot$.

Notice that the operators do not contribute in a traditional sense to the security of the key; instead they verify the soundness of the exchanges and produce a session state sst, which proves the operator's honesty to the authorities. The two operators need not agree on sst, and only one sst is required for the opening procedure (as is the case in most applications). However, if during AKE one operator validates, but not the other, then ultimately the session is aborted.

We define LIKE security in terms of three properties: key-security (KS), non-frameability (NF), and honest operator (HO). For the reader's convenience, some useful notations from the model may also be found summarized in Appendix A.

Key-Security. Our KS game extends AKE security [13]. The adversary may query all the oracles above (subject to *key-freshness* as defined below) and must distinguish a real session key from one randomly chosen from the same domain.

The KS notion is much stronger than traditional 2-party AKE. The attacker can adaptively corrupt all but the users targeted in the challenge, all but one out of the n selected authorities, and all the operators; $\mathcal{A}$ may also register malicious users, retroactively corrupt users (thus ensuring forward secrecy), and learn all but one trapdoor in the challenge session. Thus, the session key is only known by the endpoints, and by the collusion of *all* the n authorities (LI); all other (collusions of) parties fail to distinguish it from a random key. More formally, the tested target instance must be key-fresh with respect to Definition 5. The full game $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ is given in Table 1.

Definition 5 (Key Freshness). *Let π^j_{P} be the j-th created instance, associated to party $\mathsf{P} \in \mathsf{USERS}$. Let $\mathcal{A}$ be a PPT adversary against LIKE. Parse $\pi^j_{\mathsf{P}}.\mathsf{PID}$ as P' and $\pi^j_{\mathsf{P}}.\mathsf{AID}$ as $(\Lambda_i)^n_{i=1}$. The key $\pi^j_{\mathsf{P}}.\mathsf{k}$ is fresh if all the following conditions hold:*

Table 1. Games for key-security (KS, left) and non-frameability (NF, right).

$\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$:	$\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \quad \mathsf{Setup}(1^\lambda);$	$\mathsf{pp} \quad \mathsf{Setup}(1^\lambda)$
$\mathcal{O}_{\mathsf{KS}} \left\{ \begin{array}{l} \mathsf{Register}(\cdot,\cdot,\cdot), \mathsf{Send}(\cdot,\cdot), \\ \mathsf{NewSession}(\cdot,\cdot,\cdot), \\ \mathsf{Reveal}(\cdot), \mathsf{RevealTD}(\cdot,\cdot), \\ \mathsf{Corrupt}(\cdot,\cdot), \mathsf{Test}(\cdot) \end{array} \right\};$	$\mathcal{O}_{\mathsf{NF}} \left\{ \begin{array}{l} \mathsf{Register}(\cdot,\cdot,\cdot), \mathsf{Send}(\cdot,\cdot), \mathsf{Reveal}(\cdot), \\ \mathsf{NewSession}(\cdot,\cdot,\cdot), \mathsf{RevealTD}(\cdot,\cdot), \mathsf{Corrupt}(\cdot,\cdot) \end{array} \right\};$
$(i, \mathsf{P}, \mathsf{d}) \quad \mathcal{A}^{\mathcal{O}_{\mathsf{KS}}}(\lambda, \mathsf{pp});$	$(\mathsf{sst}, \mathsf{P}) \quad \mathcal{A}^{\mathcal{O}_{\mathsf{NF}}}(\lambda, \mathsf{pp});$
If $\pi^i_{\mathsf{P}}.\mathsf{k}$ is fresh and $\pi^i_{\mathsf{P}}.\mathsf{b} = \mathsf{d}$, return 1;	If $(\mathsf{A}, \mathsf{B}) \quad \mathsf{USERS}^2, n \quad \mathbb{N}, \mathsf{O} \quad \mathsf{OPS}, (\Lambda_i)^n_{i=1} \quad \mathsf{AUTH}^n$ s.t.:
Else $b' \overset{\$}{\leftarrow} \{0,1\}$, return b'.	$\quad \mathsf{Verify}(\mathsf{pp}, \mathsf{sst}, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{O.PK}, (\Lambda_i.\mathsf{PK})^n_{i=1}) = 1;$
	$\quad \mathsf{P} \quad \{\mathsf{A}, \mathsf{B}\};$
	$\quad \mathsf{P}.\gamma = 0;$
	$\quad \forall i, \text{if } \pi^i_{\mathsf{P}} \neq \quad : \mathsf{IdentifySession}(\mathsf{sst}, \pi^i_{\mathsf{P}}) = 0 \text{ or } \pi^i_{\mathsf{P}}.\alpha = 0,$
	Then return 1,
	Else return 0.

- $\pi^j_{\mathsf{P}}.\alpha = 1$, $\mathsf{P}.\gamma = 0$ *when* $\pi^j_{\mathsf{P}}.\alpha$ *became* 1*, and* $\pi^j_{\mathsf{P}}.\rho = 0$*.*
- *if* π^j_{P} *matches* $\pi^k_{\mathsf{P}'}$ *for* $k \in \mathbb{N}$*, then:* $\pi^k_{\mathsf{P}'}.\alpha = 1$*,* $\mathsf{P}'.\gamma = 0$ *when* $\pi^k_{\mathsf{P}'}.\alpha$ *became* 1*, and* $\pi^k_{\mathsf{P}'}.\rho = 0$*.*
- *if no* $\pi^k_{\mathsf{P}'}$ *matches* π^j_{P}*,* $\mathsf{P}'.\gamma = 0$*.*
- $\exists \ l \ \in \ [\![1,n]\!]$ *such that for any* $\mathsf{O} \ \in \ \pi^j_{\mathsf{P}}.\mathsf{OID}$*,* $\mathcal{A}$ *has never queried* $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda'_i)^{n'}_{i=1}, l')$ *and:*
  - $\Lambda_l.\gamma = 0$ *and* $\Lambda_l = \Lambda'_{l'}$*;*
  - $\mathsf{IdentifySession}(\mathsf{sst}, \pi^j_{\mathsf{P}}) = 1$*.*

Definition 6. *For an adversary $\mathcal{A}$ the value:*

$$\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) := \left| \mathbb{P}\left[\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

denotes its advantage *against* $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$. *A lawful-interception authenticated key-exchange scheme* LIKE *is* key-secure *if for all PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ *is a negligible function of the security parameter* λ.

Non-frameability. In this game, $\mathcal{A}$ attempts to frame a user P^* for running an AKE session (with session state sst) which P^* rejected or did not take part in. The adversary may corrupt all parties apart from P^*, as formalized in Table 1.

Definition 7. *The advantage of an adversary* $\mathcal{A}$ *in the non-frameability experiment* $\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ *in Table 1 is defined as:*

$$\mathsf{Adv}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) = \mathbb{P}\left[\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) = 1\right].$$

A lawful interception authenticated key-exchange scheme LIKE *is* non-frameable *if all PPT adversaries* $\mathcal{A}$, *have negligible* $\mathsf{Adv}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ *as a function of* λ.

Table 2. The honest-operator HO game, where q_r is the number of queries to Register, and P_i is the party input as the i-th such query.

$$
\begin{aligned}
&\underline{\mathsf{Exp}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda):} \\
&\mathsf{pp} \quad \mathsf{Setup}(1^\lambda); \\
&\mathcal{O}_{\mathsf{HO}} \quad \left\{ \begin{array}{l} \mathsf{Register}(\cdot,\cdot,\cdot), \mathsf{NewSession}(\cdot,\cdot,\cdot), \mathsf{Send}(\cdot,\cdot), \\ \mathsf{Reveal}(\cdot), \mathsf{RevealTD}(\cdot,\cdot), \mathsf{Corrupt}(\cdot,\cdot) \end{array} \right\}; \\
&(j, \mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda_i, \Lambda_i.t)_{i=1}^n) \quad \mathcal{A}^{\mathcal{O}_{\mathsf{HO}}}(\lambda, \mathsf{pp}); \\
&\text{If } \mathsf{O}.\gamma = 1 \text{ then return } \bot; \\
&\text{If } \mathsf{Verify}(\mathsf{pp}, \mathsf{sst}, \mathsf{A}.\mathsf{PK}, \mathsf{B}.\mathsf{PK}, \mathsf{O}.\mathsf{PK}, (\Lambda_i.\mathsf{PK})_{i=1}^n) = 0 \text{ then return } \bot; \\
&\text{If } \mathsf{IdentifySession}(\mathsf{sst}, \pi_{\mathsf{O}}^j.\mathsf{sid}) = 0 \text{ then return } \bot; \\
&k_* \quad \mathsf{Open}(\mathsf{pp}, \mathsf{sst}, (\Lambda_i.\mathsf{PK})_{i=1}^n, (\Lambda_i.t)_{i=1}^n); \\
&\text{Return } (k_*, \pi_{\mathsf{O}}^j, \{\mathsf{P}_i.\mathsf{PK}\}_{i=1}^{q_r}).
\end{aligned}
$$

Honest Operator. The HO game captures the fact that honest operators will abort if they detect ill-formed or non-authentic messages. The adversary must create a valid session state sst, accepted by the operators, for which the authorities Open to an incorrect session key. The attacker can provide some trapdoors and corrupt all the parties except the operator that approves sst.

Ideally, LIKE schemes should guarantee that if the (honest) operator approves a session, then the extracted key (output in $\mathsf{Exp}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$, see Table 2) will be the one used by Alice and Bob. However, malicious endpoints could run a protocol perfectly, then use a different key (*e.g.*, exchanged out of band) to encrypt messages. This flaw is universal to secure-channel establishment featuring malicious endpoints. The best a LIKE protocol can guarantee is that the extracted key is the one that *would have resulted* from an honest protocol run yielding sst. We express this in terms of a *key extractor*, which, given an operator instance and a set of public keys, outputs the key k associated with the session in which the instance took part.

Definition 8 (Key Extractor). *For any* LIKE, *a key extractor* Extract$(\cdot, \cdot)$ *is a deterministic unbounded algorithm such that, for any users* A *and* B, *operators* O_A *and* O_B, *and set of* n *authorities* $(\Lambda_i)_{i=1}^n$, *any set* {pp, A.PK, A.SK, B.PK, B.SK, O_A.PK, O_A.SK, O_B.PK, O_B.SK, k, sst, APK $= (\Lambda_i.\text{PK})_{i=1}^n, (\Lambda_i.\text{SK})_{i=1}^n, \tau_A, \tau_B, \text{PPK}$} *generated as follows:*

> pp $\leftarrow$ Setup(λ); (A.PK, A.SK) $\leftarrow$ UKeyGen(pp); (B.PK, B.SK) $\leftarrow$ UKeyGen(pp);
> $(O_A.\text{PK}, O_A.\text{SK}) \leftarrow$ OKeyGen(pp); $(O_B.\text{PK}, O_B.\text{SK}) \leftarrow$ OKeyGen(pp);
> $\forall i \in [\![1, n]\!], (\Lambda_i.\text{PK}, \Lambda_i.\text{SK}) \leftarrow$ AKeyGen(pp);
> $(\text{k}, \text{sst}_A, \text{sst}_B, \text{k}) \leftarrow$ AKE$\langle$A(A.SK), $O_A(O_A.\text{SK})$,
> $O_B(O_B.\text{SK}), \text{B(B.SK)}\rangle$(pp, A.PK, B.PK, APK);
> τ_A *is the transcript of the execution yielding* sst_A *from* O_A*'s point of view;*
> τ_B *is the transcript of the execution yielding* sst_B *from* O_B*'s point of view;*
> PPK $\leftarrow$ {$O_A.\text{PK}, O_B.\text{PK}, \text{A.PK}, \text{B.PK}$} $\cup$ {$\Lambda_i.\text{PK}$}$_{i=1}^n$;

it holds that $\forall(U, P) \in \{\text{A}, \text{B}\}^2$ *such that* $U \neq P$ *and any instance* π_{O_U} *such that* $\pi_{O_U}.\tau = \tau_U, \pi_{O_U}.\text{PID} = P, \pi_{O_U}.\text{AID} = (\Lambda_i)_{i=1}^n$, *and* $\pi_{O_U}.\text{sst} = \text{sst}$, *then:*
$\Pr[\text{Extract}(\pi_{O_U}, \text{PPK}) = \text{k}] = 1$.

   Notice that our extractor is unbounded, as it must be in order to preserve key security (otherwise the extractor would allow the operator to find the session key).

Definition 9. *For any lawful interception key-exchange scheme* LIKE *that admits a key extractor* Extract *the advantage of an adversary* $\mathcal{A}$ *in the honest-operator game* $\text{Exp}_{\text{LIKE},\mathcal{A}}^{\text{HO}}(\lambda)$ *in Table 2 is defined as:* $\text{Adv}_{\text{LIKE},\mathcal{A}}^{\text{HO}}(\lambda) =$

$$\mathbb{P}\left[\begin{array}{l} (k_*, \pi_O, \text{PPK}) \leftarrow \text{Exp}_{\text{LIKE},\mathcal{A}}^{\text{HO}}(\lambda); \\ k \leftarrow \text{Extract}(\pi_O, \text{PPK}) \end{array} : \begin{array}{l} k \neq \bot \wedge k_* \neq \bot \\ \wedge k \neq k_* \end{array}\right].$$

   A LIKE *scheme is* honest-operator secure *if, for all adversaries running in time polynomial in* λ, $\text{Adv}_{\text{LIKE},\mathcal{A}}^{\text{HO}}(\lambda)$ *is negligible as a function of* λ.

5   Our Protocol

Our LIKE schemerequires a signature scheme DS $=$ (SGen, SSig, SVer); a signature of knowledge scheme (SoK, SoKver) allowing to prove knowledge of a discrete logarithm in group $\mathbb{G}_2 = \langle g_2 \rangle$; and two NIZK proofs of knowledge – one denoted NIPoK $\{x : y = g_1^x\}$ that allows to prove knowledge of the discrete logarithm of $y = g_1^x$ in a cyclic group $\mathbb{G}_1 = \langle g_1 \rangle$ for private witness x; and another denoted NIPoK $\{x : y_1 = g_1^x \wedge y_T = g_T^x\}$ that allows to prove knowledge of the discrete logarithm of values $y_1 = g_1^x$ and $y_T = g_T^x$ in same-size groups $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_T = \langle g_T \rangle$ for private witness x.

   The proof and the signature of knowledge of a discrete logarithm can be instantiated by using Fiat-Shamir on Schnorr's protocol [39]. For the proof, we use the hash of the statement and the commitment as a challenge; for the signature of knowledge, we add

the message into the hash [17]. The proof of the discrete logarithm equality can be instantiated by using Fiat-Shamir on the Chaum and Pedersen protocol [18].

Our scheme follows the syntax in Sect. 3. We divide its presentation into four components: (a) setup and key generation, (b) authenticated key-exchange, (c) public verification, and (d) lawful interception.

Setup and Key Generation. This part instantiates the four following algorithms of the LIKE syntax presented in Sect. 3.

- Setup(1^λ): Based on λ, chooses $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T$, three groups of prime order p of length λ, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a type 3 bilinear mapping, and outputs $\mathsf{pp} = (1^\lambda, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.
- UKeyGen(pp): Runs $(\mathsf{U.PK}, \mathsf{U.SK}) \leftarrow \mathsf{SGen}(\mathsf{pp})$ and returns $(\mathsf{U.PK}, \mathsf{U.SK})$.
- OKeyGen(pp): Runs $(\mathsf{O.PK}, \mathsf{O.SK}) \leftarrow \mathsf{SGen}(\mathsf{pp})$ and returns $(\mathsf{O.PK}, \mathsf{O.SK})$.
- AKeyGen(pp): Picks $\Lambda.\mathsf{SK} \xleftarrow{\$} \mathbb{Z}_p^*$, sets $\Lambda.\mathsf{pk} \leftarrow g_1^{\Lambda.\mathsf{SK}}$ and $\Lambda.\mathsf{ni} \leftarrow \mathsf{NIPoK}\{\Lambda.\mathsf{SK} : \Lambda.\mathsf{pk} = g_1^{\Lambda.\mathsf{SK}}\}$, lets $\Lambda.\mathsf{PK} \leftarrow (\Lambda.\mathsf{pk}, \Lambda.\mathsf{ni})$, and returns $(\Lambda.\mathsf{PK}, \Lambda.\mathsf{SK})$.

The setup algorithm is run only once; key-generation is run once per party. The users and operators generate signature keys required in the authenticated key-exchange step. Authorities generate private/public keys, then prove (in zero-knowledge) that they know the private key. This prevents attacks in which authorities choose keys that cancel out other authority keys, breaking key security.
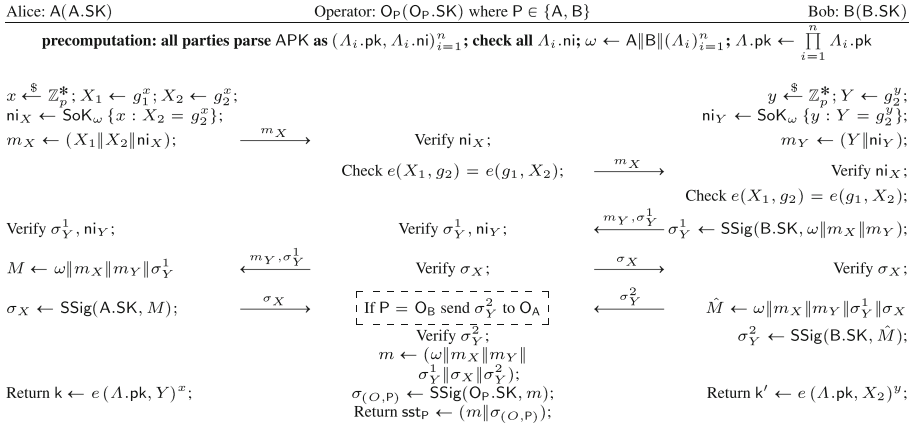

Fig. 1. The AKE component of LIKE, with operators O_A and O_B under a single heading. Operators run protocol steps in turn, forwarding messages to the next participant. If some verification fails, we assume operators instantly abort. The only message *not* forwarded by O_A to Alice is marked in the dashed box. Note that X_1 is not used during this protocol; we will see later that this element is required to generate the trapdoors.

Authenticated Key Exchange. Whenever two users communicate, they run the AKE component of our LIKE syntax, $\mathsf{AKE}\langle \mathsf{A}(\mathsf{A.SK}), \mathsf{O_A}(\mathsf{O_A.SK}), \mathsf{O_B}(\mathsf{O_B.SK}),$

$B(B.SK)\rangle$ $(pp, A.PK, B.PK, APK)$. Our protocol is described in Fig. 1. For readability we "merge" O_A and O_B since they act almost identically. Neither O_A, nor O_B computes session keys.

In Fig. 1, Alice, Bob, and the operators first verify the public keys of the n authorities, aborting if their NIZK proofs are incorrect. Notice that in the figure we omit to state that any failed verification leads to an abort. If a user has already performed these verifications, future sessions with those authority keys can proceed directly.

The heart of the protocol is the exchange between Alice and Bob. Alice generates a secret x and sends $X_1 = g_1^x, X_2 = g_2^x$, with an associated signature of knowledge linking this key share to ω. Bob proceeds similarly, sampling y and sending $Y = g_2^y$ and a signature of knowledge. The endpoints also send a signature over the transcript, thus authenticating each other and their conversation.

The two operators verify the messages they receive, aborting in case of failure, and otherwise forwarding the messages. An exception is Bob's last message, verified by both operators, but not forwarded to Alice. The operators check the signatures and signatures of knowledge and the equality of the exponents in Alice's DH elements.

Given Bob's public value Y, all the authority public keys, and her private exponent x, Alice computes her session key as a pairing of the product of all authority public keys and Y, all raised to her secret x: $k = e(\prod_{i=1}^{n} \Lambda_i.pk, Y)^x$. Due to bilinearity, this is equal to $e(\prod_{i=1}^{n} \Lambda_i.pk, g_2^y)^x = e(\prod_{i=1}^{n} \Lambda_i.pk, g_2^x)^y = e(\prod_{i=1}^{n} \Lambda_i.pk, X_2)^y$, which is Bob's key. Thus our scheme is correct for the endpoints. Note that all endpoint computations including secret values can be done by a SIM card except the pairing. Since the pairing is on public values, it can be delegated to a less secure environment (like the phone); then the exponentiation can be done on the SIM card. The operator's output is the session state sst, the signed session transcript from the their point of view.

Verification. We instantiate Verify as follows:

– Verify$(pp, sst, A.PK, B.PK, O.PK, APK) \rightarrow b$: Parse APK as a set $(\Lambda_i.PK)_{i=1}^{n}$ and parse each $\Lambda_i.PK$ as $(\Lambda_i.pk, \Lambda_i.ni)$, set $\omega \leftarrow A\|B\|(\Lambda_i)_{i=1}^{n}$. Parse sst as $\omega'\|m_X\|m_Y\|\sigma_Y^1\|\sigma_X\|\sigma_Y^2\|\sigma_O, m_X$ as $X_1\|X_2\|ni_X$ and m_Y as $Y\|ni_Y$. if:
  - $\forall i \in [\![1, n]\!]$, NIPoKver$(\Lambda_i.pk, \Lambda_i.ni) = 1$;
  - $e(X_1, g_2) = e(g_1, X_2)$;
  - SoKver$(\omega, (g_2, X_2), ni_X) = $ SoKver$(\omega, (g_2, Y), ni_Y) = 1$;
  - SVer$(B.PK, \sigma_Y^1, \omega\|m_X\|m_Y) = 1$;
  - SVer$(A.PK, \sigma_X, \omega\|m_X\|m_Y\|\sigma_Y^1) = 1$;
  - SVer$(B.PK, \sigma_Y^2, \omega\|m_X\|m_Y\|\sigma_Y^1\|\sigma_X) = 1$;
  - SVer$(O.PK, \sigma_O, \omega\|m_X\|m_Y\|\sigma_Y^1\|\sigma_X\|\sigma_Y^2) = 1$;
  then the algorithm returns 1, else it returns 0.

Intuitively, the signatures authenticate Alice, Bob, and the operator outputting sst. The Verify algorithm retraces the operator's verifications of the authorities' proofs, of the signatures and SoKs, and of the equality of the discrete logarithms for X_1 and X_2.

Lawful Interception. Lawful interception employs two algorithms, one for trapdoor generation, and another which uses all the trapdoors to extract a key.

- TDGen(pp, Λ.SK, sst):   Parse   sst   as $(\omega\|m_X\|m_Y\|\sigma_Y^1\|\sigma_X\quad\|\sigma_Y^2\|\sigma_O)$. Compute $\Lambda.t_1 \leftarrow e(X_1, Y)^{\Lambda.\text{SK}}, \Lambda.t_2 \leftarrow$ NIPoK$\left\{\Lambda.\text{SK} : \Lambda.\text{pk} = g_1^{\Lambda.\text{SK}} \wedge \Lambda.\right.$ $t_1 = e(X_1, Y)^{\Lambda.\text{SK}}\right\}$ and $\Lambda.t \leftarrow (\Lambda.t_1, \Lambda.t_2)$, and return $\Lambda.t$.
- Open(pp, sst, APK, $\mathcal{T}$): Parse $\mathcal{T}$ as $(\Lambda_i.t)_{i=1}^n$, parse sst as $\text{A}\|\text{B}\|(\Lambda_i)_{i=1}^n$ $\|m_X\|m_Y\|\sigma_Y^1\|\sigma_X\|\sigma_Y^2\|\sigma_O$, m_X as $X_1\|X_2\|\text{ni}_X$ and m_Y as $Y\|\text{ni}_Y$, APK as $(\Lambda_i.\text{PK})_{i=1}^n$, parse each $\Lambda_i.\text{PK}$ as $(\Lambda_i.\text{pk}, \Lambda_i.\text{ni})$, each $\Lambda_i.t$ as $(\Lambda_i.t_1, \Lambda_i.t_2)$ and verify the non-interactive proof of knowledge: NIPoKver $((g_1, \Lambda_i.\text{pk}, (X_1, Y), \Lambda_i.t_1), \Lambda_i.t_2)$; if any verification fails, the Open algorithm returns $\perp$. Compute and return $\text{k} \leftarrow \prod_{i=1}^n(\Lambda_i.t_1)$.

Informally, each authority Λ_i generates a trapdoor $\Lambda_i.t_1 = e(X_1, Y)^{\Lambda_i.\text{SK}}$ as well as a proof $\Lambda_i.t_2$ that ensures that $\Lambda_i.t_1$ has been generated correctly, using the same $\Lambda_i.\text{SK}$ that is associated at key-generation with that authority (*e.g.*, $\log_{g_1}(\Lambda_i.\text{pk}) = \log_{e(X_1,Y)}(\Lambda_i.t_1)$). The session key is recovered by multiplying all the trapdoors from $\Lambda_1.t_1$ to $\Lambda_n.t_1$, *i.e.*, $\prod_{i=1}^n \Lambda_1.t_i = \prod_{i=1}^n e(X_1, Y)^{\Lambda_i.\text{SK}} = \prod_{i=1}^n e(g_1^x, g_2^y)^{\Lambda_i.\text{SK}} = e(g_1, g_2^{xy})^{\sum_{i=1}^n \Lambda_i.\text{SK}} = e(g_1^{\sum_{i=1}^n \Lambda_i.\text{SK}}, g_2^{x\cdot y}) = e(\prod_{i=1}^n \Lambda_i.\text{pk}, X_2)^y$. This is indeed Bob's key, proving correctness with respect to LI.

Complexity. Due to the NIZK verification and computing the product of the n authority public keys, the AKE protocol runs in linear time with respect to n. If, however, authorities do not change between protocol runs (as is likely in practice), the runtime will be constant with respect to n.

6   Security

We present our main theorem in this section, provide proof sketches in Appendix B, and give full proofs in the full version [7].

In the following, let sid $:= X_2\|Y$, and define IdentifySession(sst, π_P^j) for some party P and integer j as follows. Parsing sst as:

$$\text{A}\|\text{B}\|(\Lambda_i)_{i=1}^n\|X_1\|X_2\|\text{ni}_X\|Y\|\text{ni}_Y\|\sigma_Y^1\|\sigma_X\|\sigma_Y^2\|\sigma_O,$$

the algorithm IdentifySession(sst, π_P^j) returns 1 iff:

- $X_2\|Y = \pi_\text{P}^i.\text{sid}$,
- if π_P^j plays the role of Alice then P $=$ A and $\pi_\text{P}^j.\text{PID} =$ B, else $\pi_\text{P}^j.\text{PID} =$ A and P $=$ B, and
- $\pi_\text{P}^j.\text{AID} = (\Lambda_i)_{i=1}^n$.

Theorem 1. *Suppose* LIKE *is instantiated with an EUF-CMA signature scheme* DS, *extractable and zero-knowledge proofs/signature of knowledge, and let* pp *be chosen such that the BDDH assumption holds. Then:*

- *Our protocol is key-secure and, for all PPT adversaries $\mathcal{A}$ making at most q_r queries to* Register, *q_ns queries to* NewSession, *q_s queries to* Send *and q_t queries to* RevealTD:

$$\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{s}}^2}{p} + q_{\mathsf{ns}} \cdot q_{\mathsf{r}}^2 \cdot \left(\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda) + q_{\mathsf{ns}} \cdot q_{\mathsf{r}} \cdot \right.$$
$$\left. \left((2 \cdot q_{\mathsf{t}} + q_{\mathsf{s}}) \cdot \epsilon_{\mathsf{SoK}}(\lambda) + q_{\mathsf{r}} \cdot \epsilon_{\mathsf{NIPoK}}(\lambda) + \mathsf{Adv}^{\mathsf{BDDH}}(\lambda)\right)\right).$$

– *Our protocol is non-frameable, and for all PPT adversaries $\mathcal{A}$ making at most q_{r} queries to* Register, *we have:* $\mathsf{Adv}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leq q_{\mathsf{r}} \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)$.
– *Our protocol is honest-operator, and for all PPT adversaries $\mathcal{A}$ making at most q_{r} queries to* Register, *we have* $\mathsf{Adv}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leqslant q_{\mathsf{r}} \cdot \epsilon_{\mathsf{NIPoK}}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)$.

An important ingredient in our protocol are the proofs and signatures of knowledge. We start with the latter. Without Alice's and Bob's signatures of knowledge, given honestly-generated X_1, X_2, Y from the challenge session, $\mathcal{A}$ could choose random values r, s, run the protocol using $X_1' \leftarrow X_1^r$, $X_2' \leftarrow X_2^r$ and $Y' \leftarrow Y^s$ for two corrupted users and the same authorities, obtain sst, and run RevealTD on sst for each authority. Using Open, the adversary obtains: $\mathsf{k}' = \prod_{i=1}^{n}(X_1', Y_2')^{\Lambda_i.\mathsf{SK}} = \left(\prod_{i=1}^{n}(X_1, Y_2)^{\Lambda_i.\mathsf{SK}}\right)^{r \cdot s}$, and can compute the targeted key as $\mathsf{k} = (\mathsf{k}')^{1 \ r \cdot s}$. This attack does not work if $\mathcal{A}$ must prove knowledge of the discrete logarithm of X_2' and Y'. Moreover, $\mathcal{A}$ cannot reuse X_2 and Y together with the signature of the honest user, since that signature uses the identity of the users as a message.

Each authority must prove knowledge of its secret key. Without this proof, an authority Λ_j could pick random r and compute: $\Lambda_j.\mathsf{pk} = g_1^r / \left(\prod_{i=1;i\neq j}^{n} \Lambda_i.\mathsf{pk}\right)$. In this case, Alice's session key becomes: $\mathsf{k} = e\left(\prod_{i=1}^{n} \Lambda_i.\mathsf{pk}, Y\right)^x = e(g_1^r, g_2^y)^x = e(X_1, Y)^r$, and the authority computes it as $e(X_1, Y)^r$. This attack is not possible if the authority must prove the knowledge of its secret key.

Finally, each authority must prove that the trapdoor is well-formed. This is essential for the HO property: otherwise, the authority can output a fake trapdoor, distorting the output of the algorithm Open.

7   Proof-of-Concept Implementation

We provide a proof-of-concept C-implementation of LIKE, which runs the entire primitive from setup to key-recovery, but omitting network exchanges. The implementation features: two authorities, Alice, Bob, and a single operator (since both operators would have to perform the same computations). The choice of two authorities was arbitrary: one for the justice system, and another for law-enforcement. More authorities would translate into a slightly higher runtime for the endpoints (one exponentiation per added authority), and a higher runtime for LI, depending on which authorities may view the session key (see also the full version [7]). However, even the worst added complexity would only involve an increased number of multiplications and potentially, the establishment of a secure multiparty channel. We measure the time-complexity of various operations during key-exchange and recovery, and provide averaged results in Table 2.

Setup. All tests are done on a Debian GNU/Linux 10 (buster) machine with an AMD Ryzen 5 1600 Six-Core Processor and 8GB RAM. We use the Ate pairing [43] over the

BN462 Barreto-Naehrig curve [29] with 128-bit security, thus following the recommendations of [9] for pairing curve parameters. We use the base points described in [29] as the generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

For the signature scheme, we use ed25519 [14], SHA-256 as our hash function, and we implement the signatures/proofs of knowledge as explained in Sect. 5. We use mcl [1] for our elliptic-curve and pairing computations. For the ed25519 signature scheme and SHA-256 we used openssl.

Results. Table 2 shows our results, averaged over 5000 protocol runs. The protocol steps involving the authorities (trapdoor generation/opening) have linear complexity in the number of authorities n ($n = 2$ in our case) because *all* trapdoors are necessary to open, and opening is linear in n. In the table, A, B, and O denote the total runtimes of Alice, Bob, and the operator respectively, during AKE (omitting the pre-computation). Verification, TDGen, and Open represent the runtime for those algorithms – which could be improved through parallelization for the last two algorithms. The runtime given for TDGen is a cumulation for the two authorities we consider; thus, each authority has a runtime of about half that value. For reference we also include the runtime of a single pairing, since it is our most expensive computation.

Operations	A	B	O	Verification	TDGen	Open	Pairing
CPU time (ms)	9.1	15.2	10.9	12.9	12.7	10.4	3.3

Fig. 2. Average CPU time (in milliseconds) for 5000 trials.

Our results above are promising, but insufficient for protocol deployment. An immediate next step would be to implement it on a mobile phone. Stronger security might even require long-term secrets for Alice and Bob to be stored in a secure element, like the SIM card, possibly delegating some computations to the mobile phone as described in Sect. 5. Moreover our protocol (and LIKE at large) would require further testing and standardization prior to deployment.

8 Conclusion

In this paper we bridge the gap between privacy and fine-grained lawful interception by introducing LIKE, a new primitive ensuring channel-establishment that is secure except with respect to: Alice, Bob, or a collusion of n legitimate authorities. In addition, users cannot be framed of wrong-doing, even by a collusion of all the authorities and operators. Finally, both the operator and the authorities are guaranteed that the LI procedure will reveal the key that Alice and Bob should have computed in any given session.

We instantiate LIKE by using efficient NIZK proofs and signatures of knowledge, and signatures. As our proof-of-concept implementation also demonstrates, the protocol is fast, and even problematic computations could be delegated on a mobile phone.

So far, our protocol only works for domestic communications, with both operators subject to an identical authority set. An extension to multiple authority sets is not obvious and we thus leave this as an open question.

A   Model Complements

Symbol	Meaning
pp	Global public parameters
Λ_i	The i-th authority
γ	Corruption bit, indicates if a party is corrupted
α	Accept bit, indicates if a party instance has terminated in an accepting state or not
ρ	Reveal bit, indicates if a party's session key was revealed
k	Session key, computed by endpoints or by collusion of authorities
sst	Session state, computed by operators
PID	Users partnered to a given instance
AID	Authorities implicitly partnered to a given instance (may open key)
OID	Operators partnered to a given instance

Fig. 3. Useful notations from our model and their significance

Definition 10 (Correctness). *Let λ a security parameter and n an integer. Run* pp $\leftarrow$ Setup(1^λ), (A.PK, A.SK) $\leftarrow$ UKeyGen(pp), (B.PK, B.SK) $\leftarrow$ UKeyGen(pp), (O_A.PK, O_A.SK) $\leftarrow$ OKeyGen(pp), (O_B.PK, O_B.SK) $\leftarrow$ OKeyGen(pp). *For all* $i \in [\![1,n]\!]$, (Λ_i.PK, Λ_i.SK) $\leftarrow$ AKeyGen(pp). *Let* APK $\leftarrow (\Lambda_i.\text{PK})_{i=1}^n$. *Then:*

- $\text{PK}_{A \to B} \leftarrow$ (pp, A.PK, B.PK, APK)*;*
- $(k_A, \text{sst}_A, \text{sst}_B, k_B) \leftarrow$
  AKE$\langle$A(A.SK), O_A(O_A.SK), O_B(O_B.SK), B(B.SK)$\rangle$($\text{PK}_{A \to B}$)*;*
- $b_A \leftarrow$ Verify(pp, sst_A, A.PK, B.PK, O_A.PK, APK)*;*
- *For all* i *in* $[\![1,n]\!]$, $\Lambda_i.t_A \leftarrow$ TDGen(pp, Λ_i.SK, sst_A)*;*
- $k_A^* \leftarrow$ Open(pp, sst_A, $(\Lambda_i.\text{PK})_{i=1}^n$, $(\Lambda_i.t_A)_{i=1}^n$)*;*
- $b_B \leftarrow$ Verify(pp, sst_B, A.PK, B.PK, O_B.PK, APK)*;*
- *For all* i *in* $[\![1,n]\!]$, $\Lambda_i.t_B \leftarrow$ TDGen(pp, Λ_i.SK, sst_B)*;*
- $k_B^* \leftarrow$ Open(pp, sst_B, APK, $(\Lambda_i.t_B)_{i=1}^n$).

For any $(b_A, b_B, k_A, k_A^*, k_B, k_B^*)$ *generated as above:* $\Pr[b_A = b_B = 1 \land k_A = k_A^* = k_B = k_B^*] = 1$.

B   Proof Sketches

Our main theorem includes three statements; we prove these in order below.

First Statement: KS. We begin by proving that the adversary has a negligible probability of winning the key-security experiment by querying the oracle Test on an instance

that matches no other instance. Notably, if the tested instance does not abort the protocol, the adversary will have to break the EUF-CMA of the signature scheme to generate the expected signatures without using a matching session.

Thus, the targeted instance must have a matching one. By key-freshness, $\mathcal{A}$ must test a key generated by two honest users, such that the trapdoor of at least one honest authority has never been queried to the oracle RevealTD. We prove (by a reduction) that $\mathcal{A}$ can only win by breaking the BDDH assumption. Let $(W_*, X_*, Y_*, W'_*, X'_*, Y'_*, Z_*)$ be a BDDH instance. We set W_* as the part of the public key $\Lambda.\mathsf{pk}$ of the honest authority, and we set X_2 as X'_*, X_1 as X_* and Y as Y'_* for the session that matches the tested instance. Then, we build the key as follows, where Λ is the honest authority: $\mathsf{k} \leftarrow Z_* \prod_{i=1;\Lambda_i \neq \Lambda}^{n} e(X_*, Y'_*)^{\Lambda_i.\mathsf{SK}}$. To compute the secret keys of the authorities controlled by the adversary, we run the extractor on the proofs of knowledge of the discrete logarithm of the public keys $\Lambda_i.\mathsf{PK}$. If Z_* is a random value, k will be random for the adversary, else $Z_* = e(X_*, Y'_*)^{\Lambda.\mathsf{SK}}$. Moreover, we simulate the oracle RevealTD on sessions with values X and Y chosen by the adversary by using the extractor on the signatures of knowledge of their discrete logarithms.

Second Statement: NF. To win the non-frameability experiment, the adversary has to build a valid session state sst for a given user, containing a valid signature of this user. We prove this theorem by reduction: assuming that an adversary is able to break the non-frameability, since this adversary generates a valid signature for a user, we can use it to break the EUF-CMA security.

Third Statement: HO. The first step of the HO proof is to design a key extractor, which takes in input a session state sst, brute-forces the discrete logarithm of Bob's Y, then computes the key as Bob would: $\mathsf{k} = e\left(\prod_{i=1}^{n} \Lambda_i.\mathsf{pk}, X_2\right)^y$. Our goal is to prove that this is the key the authorities would retrieve.

We first show (by reduction) that the adversary can only build by itself a valid sst (that may match a fake authority set) with negligible probability. Namely, if an adversary can output valid signatures for an honest operator, then we can use it to break the EUF-CMA of the signature scheme.

Moreover, for any authority Λ and any values X_1 and Y, the proof of knowledge of a trapdoor ensures that $g_1^{\Lambda.\mathsf{SK}} = \Lambda.\mathsf{pk}$ and $\Lambda.t_1 = e(X_1, Y)^{\Lambda.\mathsf{SK}}$, which implies that $\Lambda.t_1 = e(\Lambda.\mathsf{pk}, X_2)^y$ and: $\mathsf{k}_* = \prod_{i=1}^{n} \Lambda_i.t_1 = e\left(\prod_{i=1}^{n} \Lambda_i.\mathsf{pk}, X_2\right)^y$. Thus, to win the HO experiment (and return a key such that $\mathsf{k} \neq \mathsf{k}_*$), the adversary must produce a proof on a false statement, which happens with negligible probability.

References

1. MCL (2020). https://github.com/herumi/mcl
2. 3GPP: TS 33.106 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G security; Lawful interception requirements (R. 15), June 2018
3. 3GPP: TS 33.126 3GPP; Technical Specification Group Services and System Aspects; Security; Lawful Interception requirements (R. 16), September 2019
4. 3GPP: TS 33.127 3GPP; Technical Specification Group Services and System Aspects; Security; Lawful Interception (LI) Architecture and Functions (R. 16), March 2020

5. 3GPP: TS 33.128 3GPP; Technical Specification Group Services and System Aspects; Security; Protocol and procedures for Lawful Interception (LI); Stage 3 (R. 16), March 2020
6. Abelson, H., et al.: Keys under doormats. Commun. ACM **58**(10), 24–26 (2015)
7. Arfaoui, G., et al.: Legally keeping secrets from mobile operators: lawful interception key exchange (LIKE). IACR ePrint (2020). https://eprint.iacr.org/2020/684
8. Azfar, A.: Implementation and performance of threshold cryptography for multiple escrow agents in VoIP. In: Proceedings of SPIT/IPC, pp. 143–150 (2011)
9. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. J. Cryptol. **32**, 1298–1336 (2019)
10. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-48071-4_28
11. Bellare, M., Goldwasser, S.: Verifiable partial key escrow. In: CCS 1997. ACM (1997)
12. Bellare, M., Rivest, R.L.: Translucent cryptography - an alternative to key escrow, and its implementation via fractional oblivious transfer. J. Cryptol. **12**(2) (1999)
13. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48329-2_21
14. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. In: Proceedings of CHES 2011, pp. 124–142 (2011)
15. Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_3
16. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052252
17. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_5
18. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_7
19. Chen, L., Gollmann, D., Mitchell, C.J.: Key escrow in mutually mistrusting domains. In: Proceedings of Security Protocols, pp. 139–153 (1996)
20. Chen, M.: Escrowable identity-based authenticated key agreement in the standard model. Chin. Electron. J. **43**, 1954–1962 (10 2015)
21. Comey, J. (FBI) (2014). https://www.fbi.gov/news/speeches/going-dark-are-technology-privacy-and-public-safety-on-a-collision-course
22. Denning, D.E., Branstad, D.K.: A taxonomy for key escrow encryption systems. Commun. ACM **39**(3) (1996)
23. Desmedt, Y.: Abuses in cryptography and how to fight them. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 375–389. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_29
24. EU: Draft council resolution on encryption - security through encryption and security despite encryption (2020). https://files.orf.at/vietnam2/files/fm4/202045/783284_fh_st12143-re01en20_783284.pdf
25. Europol.   https://www.europol.europa.eu/newsroom/news/europol-and-european-commission-inaugurate-new-decryption-platform-to-tackle-challenge-of-encrypted-material-for-law-enforcement
26. FairTrials (2020). https://www.fairtrials.org/news/short-update-police-germany-defend-use-contact-tracing-criminal-investigations

27. Fan, Q., Zhang, M., Zhang, Y.: Key escrow scheme with the cooperation mechanism of multiple escrow agents (2012)
28. Franceschi-Bicchierai, L.: The 10 biggest revelations from Edward Snowden's leaks (2014). https://mashable.com/2014/06/05/edward-snowden-revelations/?europe=true
29. IETF: Pairing-friendly curves (2020). https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves/
30. Kahney, L.: The FBI wanted a back door to the iPhone. Tim cook said no (2019). https://www.wired.com/story/the-time-tim-cook-stood-his-ground-against-fbi/
31. Kilian, J., Leighton, F.T.: Fair Cryptosystems, revisited: a rigorous approach to key-escrow (extended abstract). In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 208–221. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_17
32. Long, Y., Cao, Z., Chen, K.: A dynamic threshold commercial key escrow scheme based on conic. Appl. Math. Comput. **171**(2), 972–982 (2005)
33. Long, Y., Chen, K., Liu, S.: Adaptive chosen ciphertext secure threshold key escrow scheme from pairing. Informatica Lith. Acad. Sci. **17**(4), 519–534 (2006)
34. Martin, K.M.: Increasing efficiency of international key escrow in mutually mistrusting domains. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 221–232. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0024467
35. Micali, S.: Fair public-key cryptosystems. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 113–138. Springer, Heidelberg. https://doi.org/10.1007/3-540-48071-4_9
36. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_22
37. Museum, C.: Clipper chip. https://www.cryptomuseum.com/crypto/usa/clipper.htm
38. Ni, L., Chen, G., Li, J.: Escrowable identity-based authenticated key agreement protocol with strong security. Comput. Math. Appl. **65**(9), 1339–1349 (2013)
39. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1989). https://doi.org/10.1007/0-387-34805-0_22
40. Shamir, A.: Partial key escrow: a new approach to software key escrow. Presented at Key Escrow Conference (1995)
41. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1984). https://doi.org/10.1007/3-540-39568-7_5
42. UN (1948). https://www.un.org/en/universal-declaration-human-rights/
43. Vercauteren, F.: Optimal pairings. IEEE Trans. Inf. Theory **56**(1), 455–461 (2010)
44. Wang, Z., Ma, Z., Luo, S., Gao, H.: Key escrow protocol based on a tripartite authenticated key agreement and threshold cryptography. IEEE Access **7**, 149080–149096 (2019)
45. Wright, C.V., Varia, M.: Crypto crumple zones: enabling limited access without mass surveillance. In: Proceedings of EuroS&P 2018. IEEE (2018)
46. Young, A.L., Yung, M.: Kleptography from standard assumptions and applications. In: Proceedings of SCN, pp. 271–290 (2010)