Pairing-free secure-channel establishment in mobile networks with fine-grained lawful interception

Xavier Bultel, Cristina Onete

Pairing-free secure-channel establishment in mobile networks with fine-grained lawful interception

Xavier Bultel[1] and Cristina Onete[2]

[1] INSA Centre Val de Loire, LIFO, Bourges, France
[2] XLIM, University of Limoges, France

Abstract. Modern-day mobile communications allow users to connect from any place, at any time. However, this ubiquitous access comes at the expense of their privacy. Currently, the operators providing mobile service to users learn call- and SMS-metadata, and even the contents of those exchanges. A main reason behind this is the Lawful-Interception (LI) requirement, by which serving networks must provide this (meta-)data to authorities, given a warrant.
At ESORICS 2021, Arfaoui *et al.* pioneered a primitive called Lawful-Interception Key-Exchange, which achieves the best of both worlds: (provably) privacy-enhanced communications, and fine-grained fine-grained, limited access to user data. Their work had two important shortcomings. First, their protocol required pairings which, while sufficiently efficient, might not always be available in the mobile setting. More importantly, that scheme was only applicable in a domestic setting, where the concerned users (Alice and Bob) were subject to the same LI authorities. The case of roaming was left as an open question.
In this paper we answer that open question. We extend the framework of Arfaoui *et al.* to allow Alice and Bob (now subject to potentially two sets of authorities) to establish a secure channel that guarantees the strong properties afforded by the LIKE schemes of ESORICS 2021. Our construction is pairing-free, faster than that of Arfaoui *et al.*, and its security relies on standard assumptions.

Keywords: Lawful interception, roaming, mobile networks, 5G

1 Introduction

End-to-end secure communication is one of cryptography's most fundamental and researched topics. In short, secure-channel establishment allows peers Alice and Bob to exchange messages over an open line, such that: no one but themselves can learn anything about the contents of those messages (confidentiality); they can verify the authorship of those messages (authenticity); even if their long-term secrets leak, past communications remain confidential and authentic (perfect forward secrecy – PFS). In the wake of Edward Snowden's revelations of mass surveillance on civilian conversations, these properties are particularly important.

People's awareness of and desire for the privacy of their communications (leading, *e.g.*, to the adoption of applications advertizing privacy, like WhatsApp or Viber) is now stronger than ever. Commendable regulations, such as the EU's General Data Protection Regulation (GDPR[3]) or the new electronic privacy regulation (ePR [4]) have revolutionized today's Web-browsing by limiting the personal data collected by clients and giving users the right to remove information which may harm them.

Yet, at the opposite end, law-enforcement agencies and governments have, for years, been pushing for *less* privacy in communications. We are often told this is in the name of (inter-)national security, preventing horrific crimes, such as child abuse, terrorism, or organ trafficking. Thus, while law-enforcement agencies advocate for privacy in communications, they also want "back doors": ways to exceptionally access data in order to prevent crime.

A recent, iconic case is that of FBI versus Apple[5], in which the FBI demanded a back door allowing them access to locked Apple phones. Apple refused, explaining that such back doors were unconstitutional, and that the resulting unrestrained access would set a dangerous precedent.

[3] See `https://gdpr-info.eu/` for the full text.
[4] See `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX\%3A52017PC0010`.
[5] See `https://epic.org/amicus/crypto/apple` for a summary of that debate.

Yet, even staunch defenders of privacy like Abelson *et al.* [1] agree that limited, well-regulated access to data may be useful and acceptable (when *e.g.*, used to prevent the heinous crimes cited above). Whereas *universal* back doors do not discriminate between subpoenaed and un-subpoenaed sensitive information, *fine-grained, exceptional* access to data can be viewed as constructive.

The generic notion of *key-escrow* captures just such a fine-grained back door that targets a (set of) key(s). Despite extensive research, key-escrow has many limitations, due mostly to its wide-ranging application scenarios. For instance, it may be hard to "oblige" Alice and Bob to use a particular channel (with escrowed keys); moreover, many key-escrow protocols require parties that recover keys (the authorities) to be always online.

At ESORICS 2021 Arfaoui *et al.* introduced Lawful Interception Key-Exchange protocols (LIKE) as a way to provide punctual exceptional access to mobile conversations between users, while preserving the privacy of all users and conversations not targeted by a subpoena [3]. This scenario, which naturally obliges Alice and Bob to converse over a specific channel via their serving networks, also comes with enforced security properties (non-frameability and honest operator) and does not require the authorities to be online.

The LIKE protocol of [3] can work with multiple service providers, but not multiple sets of authorities. In other words, in contrast to LI requirements during roaming, the scheme of [3] cannot guarantee LI access in both Alice's country and Bob's country. This limits the use of that protocol to domestic communications (featuring only one set of authorities) or to roaming scenarios for which one country agrees to forego its right of LI in favour of another country. As a second disadvantage, the protocol of [3] requires pairings, which require specialized implementation libraries which might not always be readily available.

In this paper, we extend LIKE protocols to capture both domestic and roaming communications. Although this opens exceptional access from one to two sets of authorities, we note that this is a legal *requirement* of international communications. Our protocol, which is pairing-free and very fast, maximizes privacy within the bounds of the law, guaranteeing the same, strong properties pioneered by [3], but with respect to two independent set of authorities.

The LIKE setup. Precisely defined by law and 3GPP standards, Lawful interception (LI) is the process by which exceptional access is given to a set of *authorities*, in possession of a warrant, to mobile data or metadata pertaining to a *user*.

In current architectures, when two mobile-users Alice and Bob decide to communicate, the data they share is forwarded by the operators serving the users. This allows operators to answer LI queries for (meta-)data of users, when queried by an accepted set of authorities. Unfortunately, the operators learn much more than just the data they forward, and are given access to enormous amounts of potentially-sensitive data. This damages user privacy and makes operators a target to entities wanting access to that data.

Arfaoui *et al.* present in [3] a pairing-based alternative to this approach, in which Alice and Bob establish an end-to-end-secure channel that denies operators access to their data. Exceptional LI access is guaranteed by bilinearity, since Alice and Bob "embed" the product of a number of authority keys into the session key. During LI, each authority generates a trapdoor to the session key; by then using bilinearity and the set of all the trapdoors, the authorities recover the key.

A clear downside of the protocol is Alice and Bob *must* embed the same set of authorities into the session key, or compute distinct session keys. Thus, [3] leaves LI in the presence of two sets of authorities as an open question.

Our protocol. We consider a LIKE setup in which Alice, currently in country A, is receiving mobile service from operator $\mathsf{O_A}$. Bob is in country B, receiving mobile service from another operator $\mathsf{O_B}$. When Alice and Bob communicate over the mobile network, $\mathsf{O_A}$ and $\mathsf{O_B}$ forward their data, but (as in [3]) in encrypted form. Unlike [3], we consider that Alice's communication will be subject to a set of authorities in country A, while Bob's may be opened by a different set of authorities, in country B.

In our protocol, users Alice and Bob (essentially) use signed Diffie-Hellman to establish their secure channel. The authority keys are no longer embedded in the session key; instead, the two endpoints enable LI by encrypting the session key with a function of the authority keys (akin to ElGamal encryption), providing the ciphertext and a proof of its well-formedness to the operators. The ciphertext and proof are *only received and verified by the operator providing service to that endpoint.* Our protocol's security relies on standard assumptions: the hardness of the Decisional Diffie-Hellman problem, the unforgeability

of the signature scheme, and the security of simple proofs and signatures of knowledge. Our protocol is also fast, as we show through a proof-of-concept implementation described in Section 5.

Other related work. Though closely related to [3], our work makes important original contributions: we extend LIKE to handle two different sets of authorities (an open question in [3]), and provide a provably-secure, pairing-free construction achieving it.

We provide stronger guarantees than typical key-escrow [19, 12, 4, 23, 21, 6, 10, 17, 15, 16, 13, 18, 22, 20, 11]. First, our chosen use-case (mobile communications) *obliges* participants to communicate via two serving networks which can now vouchsafe for Alice's and Bob's protocol-compliance. Moreover, our protocol can handle two independent sets of authorities simultaneously, one performing LI on Alice's end, and the other, on Bob's. Finally we guarantee Arfaoui *et al.*'s strong properties: fine-grained, session-based LI access; no central key-generation authority; no trust required in authorities; and finally, we do not require those authorities to be online except at LI opening.

A parallel line of research to ours [7, 24] aims to make LI possible, but computationally very expensive. Unfortunately, that approach contravenes the 3GPP requirement of a *timely* return of the subpoenaed content. This timeliness requirement affects the operator directly, creating an incentive for it to only adopt solutions that allow for speedy key-recovery.

2 Preliminaries

Notations. Throughout the paper, $\lambda \in \mathbb{N}$ will denote a security parameter. The notation $x \leftarrow y$ signifies that a variable x is assigned a value y. We write $x \xleftarrow{\$} A$ when x is sampled identically and uniformly at random from the set A, and, for an algorithm Alg, the notation $y \leftarrow \mathsf{Alg}(x)$ expresses the fact that, if run on input x, Alg outputs y.

Definition 1 (Decisional Diffie-Hellman (DDH) assumption). *Let $(\mathbb{G}, p, g)$ be a prime order group. Then DDH holds when for any random tuple $(x, y, z) \in (\mathbb{Z}_p^*)^3$, no probabilistic polynomial time (PPT) adversary can distinguish between $(g^x, g^y, g^{x \cdot y})$ and (g^x, g^y, g^z) with non-negligible probability in $|p|$.*

Definition 2 (Digital signatures). *A digital signature scheme* DS *is a triplet* $(\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ *such that* SGen *takes in input* 1^λ *and outputs a pair* $(\mathsf{PK}, \mathsf{SK})$*,* SSig *takes in input a key* SK *and a message m and outputs a signature σ, and* SVer *takes as input a key* PK*, a message m, and a signature σ and outputs* 1 *if the signature is deemed valid, and* 0 *otherwise. A digital signature is* existentially unforgeable against chosen message attacks *(EUF-CMA) if no PPT adversary interacting with a signing oracle is able to forge a fresh and valid signature.*

(Non-Interactive) Proof/Signatures of Knowledge [5, 9] (NIPoK and SoK) allow a prover to convince a verifier that it knows a *witness* to a particular *statement*, without revealing information about that witness. In Sok, the prover embeds the message to sign in the proof. We use the Camenisch/Stadler notation [8] to formalize both these primitives below.

Definition 3 (Signature/Proof 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* SoK *for $\mathcal{L}$ is a pair of algorithms* $(\mathsf{SoK}, \mathsf{SoKver})$ *such that* $\mathsf{SoK}_m \{w : (s, w) \in \mathcal{R}\}$ *outputs a signature π and* $\mathsf{SoKver}(m, s, \pi)$ *outputs* 1 *for acceptance,* 0 *otherwise. The formal definition of NIPoK follows along similar lines – except NIPoKs do not require messages. In addition, a Zero-Knowledge (ZK) proof/signature of knowledge allows no additional information about the witness to leak.*

3 LIKE protocols

Originally described by Arfaoui *et al.* [3], LIKE protocols are principally meant to allow two *users* Alice (denoted A) and Bob (denoted B), whose mobile communication is forwarded by *operators* $\mathsf{O_A}$ (providing mobile service to A) and $\mathsf{O_B}$ (serving B), to establish a secure channel that can be exceptionally opened by some *authorities*.

In the original setting of [3], at each connection Alice, Bob, and the operators must agree on the same subset of authorities. While this might be realistic for domestic mobile communications, it is not so for the case of roaming, in which Alice and her serving network are in a different country (thus affiliated to different authorities) than Bob and his serving network. In this work we extend the work of Arfaoui *et al.* to also cover roaming – an extension which will require small modifications to the primitive's syntax and security model.

Context and parties. Like [3], we consider a set of user USERS, a set of operators OPS, and a set of authorities AUTH (the latter will include the set of all possible authorities, from all possible countries). Anticipating our setup, note that each protocol session will take place in the presence of two subsets of authorities, which might, or might not, have a non-void intersection.

The union $\mathsf{USERS} \cap \mathsf{OPS} \cap \mathsf{AUTH}$ is the set of parties. It is assumed that the user set is disjoint from the union of the other two sets: in other words, a user can never be an authority or an operator. Just like Arfaoui *et al.*, we also assume that technically the operator and authority sets are disjoint. However, note that in reality, an operator can in fact also play a part in exceptional opening – in that case, we "split" the same moral identity into two separate ones which have knowledge of each other's keys: one entity plays the role of the operator, and the other, the role of the authority.

Protocol structure and intuition. Like [3] we envision LIKE protocols as a sequence of steps: setup, key-generation, authenticated key-exchange, verification, and lawful interception (the latter step being made up of two algorithms). During setup, we generate some common public parameters (such as the description of a prime-order group). Then, each type of party (users, operators, and authorities) will use its own key-generation algorithm to generate credentials. These first two steps are depicted in Figure 1, and while we recall their syntax below, it is unchanged compared to [3].

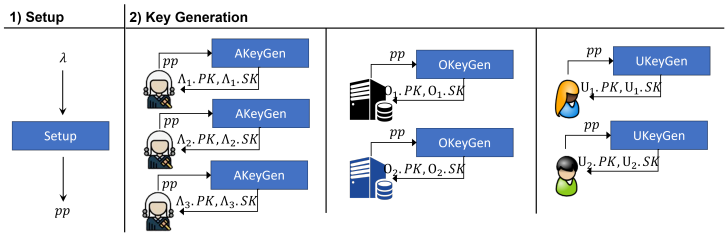

Fig. 1. The Setup and Key-Generation steps, for two users, two operators, and three authorities

Once parameters are generated, two users and their respective serving networks engage in an authenticated key-exchange protocol (as depicted in Figure 2), requiring Alice and her serving operator to agree on one subset of authorities, while Bob and his operator must agree on a (possibly distinct) set of authorities. This is reflected in a slight modification of the syntax of our authenticated key-exchange algorithm below.

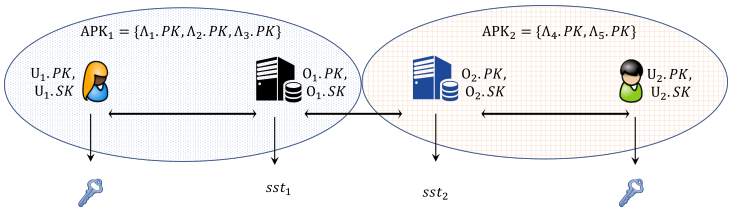

Fig. 2. Authenticated key-exchange. The two left parties agree on a subset of authorities (3 in this example), while the two right parties agree on a different set of two authorities. The operators compute a session state, while the parties compute a session key.

During authenticated key-exchange, the two endpoints compute a session key, while the operators verify the fact that the transcript is protocol compliant, and finally each output a session state. The

syntax of [3] is permissive, allowing the two session states, corresponding to the two operator outputs in a single session, to differ. In the case of roaming (for two different sets of authorities), those states will always differ, since they include the authority public-key sets.

Each session state is publicly verifiable with respect to the validity of the transcript and the identities of both the participants that generated it, and the authorities that were considered. Once more, the syntax of [3] can remain unchanged, although in practice we must verify the session state output by $\mathsf{O_A}$ with respect to the authority set considered by A and her operator, while the session state output by $\mathsf{O_B}$ is verified with respect to the authority set considered by B and his operator.

The final step is lawful interception, which consists of two algorithms. Using the session state output by one of the operators, the authorities with respect to which that session state was established will first generate some trapdoors. Using all the trapdoors in input, the opening algorithm will then yield the session key established by Alice and Bob (as depicted in Figure 3).

Notations. In both the formalization below and the model, we use dot notations to separate specific attributes of a party from the party owning it. For instance $\mathsf{A.PK}$ denotes Alice's public key, and $\Lambda_i.\mathsf{SK}$ is the secret key of the i-th authority. The two operators are denoted $\mathsf{O_A}$ and $\mathsf{O_B}$ because they provide service to Alice and Bob respectively[6]. Parties A and $\mathsf{O_A}$ agree to run the protocol in the presence of a subset of n_A authorities, while parties B and O agree to run it in the presence of a possibly different subset of n_B authorities. We note that the choice of authorities is thus limited to one of the two sides of the protocol.

Extending the LIKE definition. The only extension we make to the already-existing LIKE syntax of [3] is to provide for two sets of authorities in the AKE protocol. The new formal syntax of this algorithm is as follows:

- $\mathsf{AKE}\langle\mathsf{A}(\mathsf{A.SK}), \mathsf{O_A}(\mathsf{O_A.SK}), \mathsf{O_B}(\mathsf{O_B.SK}), \mathsf{B}(\mathsf{B.SK})\rangle(\mathsf{PK_{A\to B}}) \to (\mathsf{k_A}, \mathsf{sst_A}, \mathsf{sst_B}, \mathsf{k_B})$: An authenticated key-exchange protocol between users $(\mathsf{A}, \mathsf{B}) \in \mathsf{USERS}^2$ and operators $(\mathsf{O_A}, \mathsf{O_B}) \in \mathsf{OPS}^2$. The parties each take as input a secret key, and have access to the same set of public values $\mathsf{PK_{A\to B}}$ containing: parameters pp, public keys $(\mathsf{A.PK}, \mathsf{B.PK})$, and **two distinct vectors of authority public keys** $(\mathsf{APK_A}, \mathsf{APK_B}) = ((\Lambda_i^\mathsf{A}.\mathsf{PK})_{i=1}^n, (\Lambda_i^\mathsf{B}.\mathsf{PK})_{i=1}^m)$ with $\Lambda_i^\mathsf{A} \in \mathsf{AUTH}$ for all $i \in [\![1, n]\!]$ and $\Lambda_i^\mathsf{B} \in \mathsf{AUTH}$ for all $i \in [\![1, m]\!]$. Note that we place no restrictions on whether or not the same public key exists in both vectors. We also note that, while the syntax seems to require Alice to know the keys of the authorities on Bob's side, this is not really necessary for our protocol (Alice will never need those). In that sense, the use of a universal set $\mathsf{PK_{A\to B}}$ is only for convenience and legibility. At the end of the protocol, A (resp. B) returns a *session secret key* $\mathsf{k_A}$ (resp. $\mathsf{k_B}$) and the operator $\mathsf{O_A}$ (resp. $\mathsf{O_B}$) returns a (public) *session state* $\mathsf{sst_A}$ (resp. $\mathsf{sst_B}$). In case of failure, the parties output a special symbol $\perp$ instead.

We make no modification to the remaining algorithms in the original framework of Arfaoui *et al.* [3]. However, for completeness, we recall them here.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Generates public system parameters pp.
- $\mathsf{UKeyGen}(\mathsf{pp}) \to (\mathsf{U.PK}, \mathsf{U.SK})$: Used by mobile users to generate a public/private *user key pair*.
- $\mathsf{OKeyGen}(\mathsf{pp}) \to (\mathsf{O.PK}, \mathsf{O.SK})$: Used by operators to generate a public/private *operator key pair*.
- $\mathsf{AKeyGen}(\mathsf{pp}) \to (\Lambda.\mathsf{PK}, \Lambda.\mathsf{SK})$: Used by authorities (irrespective of country) to generate a public/private *authority key pair*. Note that, although we place no restrictions on the subsets of authorities used, they *must* use the same public system parameters (at least, the same per AKE session), regardless of their country in order to allow the protocol to work.
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{sst}, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{O.PK}, \mathsf{APK}) \to \mathsf{b}$: Verifies if an input session state sst, is consistent with a session having been correctly run and authenticated by input operator O between users A and B for a set of authorities APK. If the verification succeeds the algorithm outputs 1.
- $\mathsf{TDGen}(\mathsf{pp}, \Lambda.\mathsf{SK}, \mathsf{sst}) \to \Lambda.\mathsf{td}$: Generates a trapdoor $\Lambda.\mathsf{td}$ for a session state sst, using authority secret key $\Lambda.\mathsf{SK}$.
- $\mathsf{Open}(\mathsf{pp}, \mathsf{sst}, \mathsf{APK}, \mathcal{T}) \to \mathsf{k}$: Recovers a purported key for session with state sst, given authority public keys $\mathsf{APK} = (\Lambda_i.\mathsf{PK})_{i=1}^n$ and trapdoors $\mathcal{T} = (\Lambda_i.\mathsf{td})_{i=1}^n$. If no key can be recovered, the output is $\perp$.

[6] Note that this does not necessarily mean that Alice and Bob subscribe to those two particular operators.

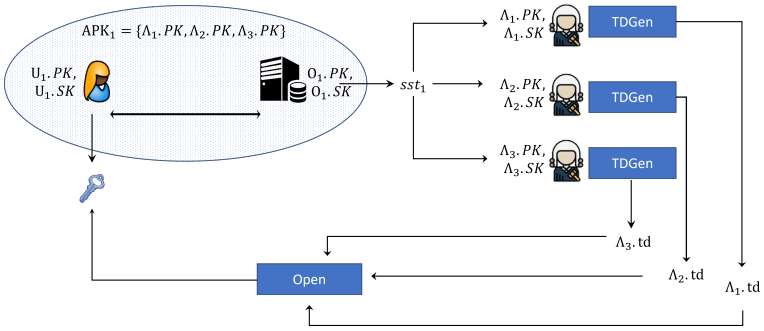

Fig. 3. Lawful interception on Alice's side of the conversation. Note that, to recover the key, all the authorities stipulated during key-exchange must generate and use their trapdoors.

4 Our protocol

For our scheme, we assume that both the operators and the users employ a digital signature scheme $\mathsf{DS} = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ – namely, their long-term credentials will be used for signing messages. We will require two types of zero-knowledge proofs of knowledge:

- **DLog:** The first type of ZK NIPoK we require is a proof of knowledge of a discrete logarithm. We use this to prove that the authorities generated their long-term keys correctly; this is crucial in order to guarantee that exceptional opening occurs only when all the authorities in the session state's authority vector cooperate. We thus actively fine-grain lawful interception and prevent easy mass surveillance.
- **Equal exponent:** The second type of ZK NIPoK we need is a proof of equality of discrete logarithms. We use this in the trapdoor-generation algorithm, for which authorities need to prove that their trapdoors were generated using the same exponent as was used in their long-term credentials. This prevents malicious authority behaviour which might yield a different session key when opening than the key computed by Alice and Bob.

The signature of knowledge will also prove a knowledge of equal exponents, which assures the operator that Alice's and Bob's messages will allow the authorities to recover the key yielded by that protocol session. Note that in our protocol Alice's operator will check protocol compliance with respect to the set of authorities in Alice's country, and Bob's operator will check compliance with respect to the authorities in Bob's country.

In the following, we detail our protocol in terms of the syntax presented in the previous section.

Setup and Key Generation. As opposed to the protocol of Arfaoui *et al.* [3], we do not require pairings. During setup we just fix the description of a group $\mathbb{G}$ of prime order p with some generator g. Both users and operators use the key-generation algorithm of the signature scheme in order to generate their credentials, whereas authorities generate an exponent in $\mathbb{Z}_p^*$ as their secret key, publish the public key corresponding to that exponent, and prove in zero-knowledge that the keys are well-formed.

- $\mathsf{Setup}(1^\lambda)$: Based on λ, choose $\mathbb{G}$ a group of prime order p, a generator of g of $\mathbb{G}$, and output $\mathsf{pp} = (\lambda, \mathbb{G}, p, g)$.
- $\mathsf{UKeyGen}(\mathsf{pp})$: Run $(\mathsf{U.PK}, \mathsf{U.SK}) \leftarrow \mathsf{SGen}(1^\lambda)$ and return $(\mathsf{U.PK}, \mathsf{U.SK})$.
- $\mathsf{OKeyGen}(\mathsf{pp})$: Run $(\mathsf{O.PK}, \mathsf{O.SK}) \leftarrow \mathsf{SGen}(1^\lambda)$ and return $(\mathsf{O.PK}, \mathsf{O.SK})$.
- $\mathsf{AKeyGen}(\mathsf{pp})$: Pick $\Lambda.\mathsf{SK} \xleftarrow{\$} \mathbb{Z}_p^*$, compute $\Lambda.\mathsf{pk} \leftarrow g^{\Lambda.\mathsf{SK}}$ and $\Lambda.\mathsf{ni} \leftarrow \mathsf{NIPoK}\left\{\Lambda.\mathsf{SK} : \Lambda.\mathsf{pk} = g^{\Lambda.\mathsf{SK}}\right\}$, set $\Lambda.\mathsf{PK} \leftarrow (\Lambda.\mathsf{pk}, \Lambda.\mathsf{ni})$. Return $(\Lambda.\mathsf{PK}, \Lambda.\mathsf{SK})$.

Authenticated key-exchange. The protocol AKE is instantiated as described in Figure 4.

This protocol presents a crucial difference in the key-computation compared to [3]. Beyond being pairing-free, our scheme no longer embeds the authorities into the session key, thus allowing two distinct sets of authorities to open it.

As described in Figure 4, we require everyone to be sure, when using a particular authority public key, that it was generated according to protocol. This is why we have a precomputation phase in which the users and operators check the public keys of the authorities they are about to use. However, we note

that this precomputation phase can be reused across sessions: once verified, a public key need never be verified again.

During the same precomputation phase, A and O_A multiply the public keys of the involved authorities, thus obtaining an auxiliary value h_A (and respectively h_B).

Our protocol relies on signed Diffie-Hellman, with the key being $X^y = Y^x = g^{xy}$, within the group $\mathbb{G}$. However, we also embed some elements into the protocol, which will later enable authorities on both sides to recover trapdoors to the AKE session. These additional elements are only exchanged between the user and its serving network; once they are verified, they are added by the operator to the session state, but not forwarded to the other operator. This is why the transcript of the protocol between O_A and O_B is that of the signed Diffie-Hellman protocol, while the one between A and O_A (and B and O_B respectively) contains additional elements.

In order to allow the authorities in APK_A to recover the key, user A "encrypts" the session key (Y^x) with h_A^x, obtaining a ciphertext H_A. Looking ahead, in order to "neutralize" h_A and recover the key we require the contributions of all the parties whose public keys are in h_A. Bob will do the same on his side, but for the authorities in APK_B and respectively for h_B; the ciphertext obtained is H_B.

Note that successful session opening depends on the well-formedness of the ciphertexts H_A and H_B. In order to prevent the users from cheating, we require them to provide a signature of knowledge on a message consisting of the identities of Alice, Bob, and the authorities, proving in zero-knowledge that they have used the same exponent in computing their H value and for their Diffie-Hellman key-share.

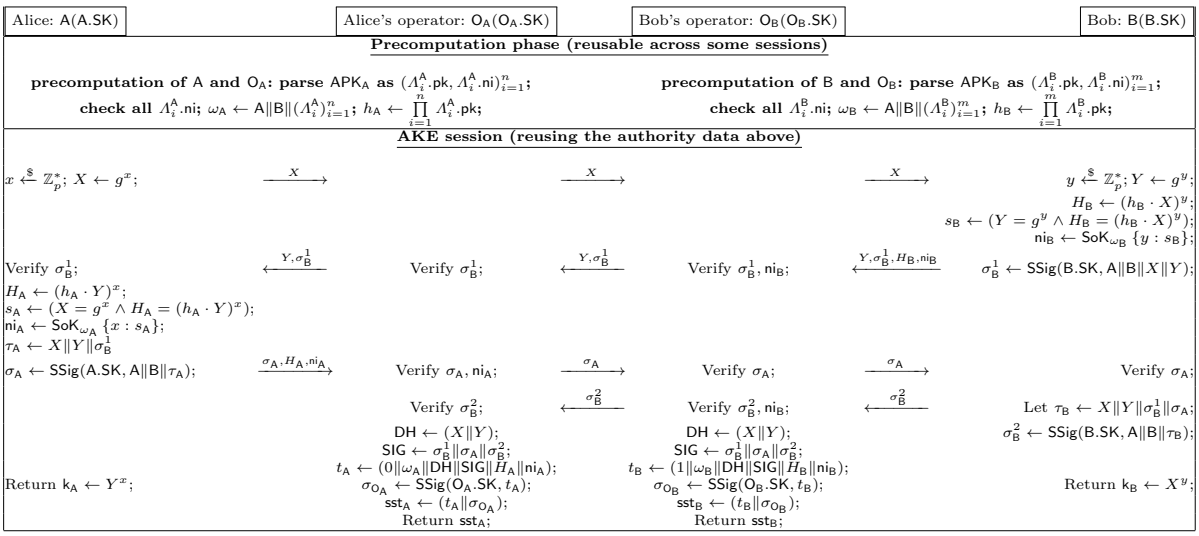

Fig. 4. The authenticated key-exchange step of our protocol $\mathsf{AKE}\langle\mathsf{A}(\mathsf{A.SK}), \mathsf{O}_\mathsf{A}(\mathsf{O}_\mathsf{A}.\mathsf{SK}), \mathsf{O}_\mathsf{B}(\mathsf{O}_\mathsf{B}.\mathsf{SK}), \mathsf{B}(\mathsf{B.SK})\rangle(\mathsf{PK}_{\mathsf{A}\to\mathsf{B}})$.

Notice that at no point in the execution of the protocol does Alice need to know the identities or public keys of Bob's authorities, nor vice-versa. This is a crucial feature of the protocol, which is essential in real life, since often the authorities in one country do not (and sometimes should not) know the identities of the relevant LI authorities in another country.

Verification. We note that verification is unilateral, in the sense that either we verify the session state on Alice's side of the conversation with respect to the identities of the participants and the authorities on her side, or we do the same on Bob's side, with respect to the authorities and session state on his side.

- $\mathsf{Verify}(\mathsf{pp}, \mathsf{sst}, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{O.PK}, \mathsf{APK}) \to b$: Parse APK as a set $(\Lambda_i.\mathsf{PK})_{i=1}^n$ and parse each $\Lambda_i.\mathsf{PK}$ as $(\Lambda_i.\mathsf{pk}, \Lambda_i.\mathsf{ni})$, set $\omega \leftarrow \mathsf{A}\|\mathsf{B}\|(\Lambda_i)_{i=1}^n$ and $h \leftarrow \prod_{i=1}^n \Lambda_i.\mathsf{pk}$. Parse sst as $d\|\omega'\|X\|Y\|\sigma_\mathsf{B}^1\|\sigma_\mathsf{A}\|\sigma_\mathsf{B}^2\|H\|\mathsf{ni}\|\sigma_O$ and set $Z_0 = X$ and $Z_1 = Y$. If $\sigma_\mathsf{A}, \sigma_\mathsf{B}^1, \sigma_\mathsf{B}^2$, and σ_O are valid signatures, ni and $\Lambda_i.\mathsf{ni}$ for all $i \in [\![1, n]\!]$ are valid signatures/proofs of knowledge, and $\omega = \omega'$, then the algorithm returns 1, else it returns 0.

Lawful interception. Lawful interception consists of two stages. First, in an individual effort, each authority computes a trapdoor to the session key. Then, during opening, the trapdoors must be combined to retrieve the session key from the ciphertext provided by the endpoint users (either H_{A} or H_{B}, as the case may be).

For Alice's side, each authority computes as a first element of the trapdoor the group element $X^{\Lambda.\mathsf{SK}}$ (where X is Alice's key-exchange element). The second element of the trapdoor is a proof of well-formedness from the part of the authority, namely demonstrating that the same private key was used to compute both the authority's long-term public key and the authority's first trapdoor element. The same is true for Bob, except that the authorities will compute $Y^{\Lambda.\mathsf{SK}}$.

The opening procedure begins with a verification of both the validity of the session state with respect to the expected participants and authorities, and soundness of the handshake, and a verification of the well-formedness of all the trapdoors used in input to the opening algorithm. On Alice's side, the session key is obtained from the ciphertext H_{A}, by dividing it by the product of the trapdoors obtained by all the authorities agreed upon by Alice and O_{A}. On Bob's side, the procedure is identical, using H_{B} and the authorities on Bob's side.

- $\mathsf{TDGen}(\mathsf{pp}, \Lambda.\mathsf{SK}, \mathsf{sst})$: Parse sst as $(d\|\omega'\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}\|\sigma^2_{\mathsf{B}}\|H\|\mathsf{ni}\|\sigma_O)$ and set $Z_0 \leftarrow X$, $Z_1 \leftarrow Y$, $\Lambda.\mathsf{td}_1 \leftarrow Z_d^{\Lambda.\mathsf{SK}}$, $\Lambda.\mathsf{td}_2 \leftarrow \mathsf{NIPoK}\left\{\Lambda.\mathsf{SK} : \Lambda.\mathsf{PK} = g^{\Lambda.\mathsf{SK}} \wedge \Lambda.\mathsf{td}_1 = Z_d^{\Lambda.\mathsf{SK}}\right\}$ and $\Lambda.\mathsf{td} \leftarrow (\Lambda.\mathsf{td}_1, \Lambda.\mathsf{td}_2)$, and returns $\Lambda.\mathsf{td}$.
- $\mathsf{Open}(\mathsf{pp}, \mathsf{sst}, \mathsf{APK}, \mathcal{T})$: Parse $\mathcal{T}$ as $(\Lambda_i.\mathsf{td})_{i=1}^n$, sst as $d\|\omega'\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}\|\sigma^2_{\mathsf{B}}\|H\|\mathsf{ni}\|\sigma_O$, set $Z_0 = X$ and $Z_1 = Y$, parse APK as $(\Lambda_i.pk)_{i=1}^n$, parse each $\Lambda_i.\mathsf{PK}$ as $(\Lambda_i.\mathsf{pk}, \Lambda_i.\mathsf{ni})$, each $\Lambda_i.\mathsf{td}$ as $(\Lambda_i.\mathsf{td}_1, \Lambda_i.\mathsf{td}_2)$ and verify that $\mathsf{NIPoKver}((g, \Lambda_i.\mathsf{pk}, Z_d, \Lambda_i.\mathsf{td}_1), \Lambda_i.\mathsf{td}_2) = 1$; if any verification fails, the Open algorithm returns $\perp$. Compute and return $\mathsf{k} \leftarrow \frac{H}{\prod_{i=1}^n (\Lambda_i.\mathsf{td}_1)}$.

Once more we draw the reader's attention to the fact that the opening procedure only requires the trapdoors of the set of authorities stipulated in the given session state. In other words, if O_{A} outputs a session state $\mathsf{sst}_{\mathsf{A}}$ that is then used for the lawful interception steps, then the opening does not require the contribution of the authorities on Bob's side of the conversation.

The flexibility of our approach. We return here to some of the desirable features of LIKE schemes that we mentioned in Section 1. We have already discussed one of them: the complete independence of the LI processes on either side of the communication: Alice and her operator need know nothing about the lawful-interception process (including the names or public keys of the authorities) on Bob's side of the communication. There is also no cardinality requirement on the sizes of the two authority subsets: in other words, Alice's communication might be subject to more authorities than Bob's. We also note that Alice and her operator are never involved by the authorities on Bob's side, should Bob's side of the communication be subject to LI requirements (nor vice-versa). These properties are all novel to our scheme, making it much more useful (and realistic) in practice than previous work.

Our scheme also benefits from the flexibility of Arfaoui *et al.*'s protocol. The separation of the party set into disjoint sets (authorities, operators, and users) is not as restrictive as it appears. This separation is more about cryptographic data and ownership of keys. Thus, if in a given country an operator is, in fact, an authority, then two sets of keys are generated, and the operator uses its operator keys while functioning as an operator, and its authority keys for LI queries. If only *some* of the authorities need to recover the secret, then those authorities can create a secure channel and use that to publish their trapdoors (while the remaining authorities, who need not recover the secret, can just publish it outside that secure channel).

5 Implementation results

We implement a prototype of our protocol and the protocol of Arfaoui *et al.* [3] using the Charm-Crypto framework [2]; our code is available at [14]. Since [3] requires pairings, we need to use a pairing-friendly curve for the initial comparison – in our case, `MNT159` with 159-bit base field from the PBC library, which ensures a security level of 80 bits and has fairly-fast pairing computations. However, our protocol does not, in fact, require pairings; thus, we can also run our protocol using the `NIST/X9.62/SECG` curve over a 192 bit prime field `prime192v` – for which our protocol gains a significant performance leap. We note that by using the second curve, we gain in security, as well as efficiency.

We use Schnorr-like protocols and signatures to instantiate the zero-knowledge proofs of knowledge, the signatures of knowledge, and the signature schemes. We run our script on `Ubuntu 18.04.4 LTS (64 bits)` using an `Intel Core i5-8365U CPU @ 1.60GHz×8` processor.

In Figure 5 we summarize our implementation results, comparing the protocol of Arfaoui *et al.* [3] and our protocol. We evaluate the computation cost of the precomputation phase, the key-exchange protocol for one entity (Alice, Bob or an operator), the active phase of the key exchange protocol (where the protocol steps are executed sequentially), the verification of `sst`, the trapdoor generation, and the extraction of the key. The first part of the table in Fig. 5 evaluates the number of exponentiations in the prime order groups and, optionally, the number of pairing-computations, which are the most costly operations of the protocols. Note that pairings are, in general, much less efficient than exponentations.

The second part of the table in Fig. 5 evaluates the execution time of the two protocols (for $n = 3$ authorities)[7]. We evaluate the performance of our protocol for the same setting as for Arfaoui *et al.* [3] (`MNT159`), and then showcase its true potential by using the more appropriate `prime192v` curve. The lack of pairings provides an evident advantage in the `prime192v` setting, but is even present on `MNT159` curves: indeed, on `MNT159` the pairing takes input of the type $(\tilde{g}, \tilde{h})$ from $\mathbb{G}_1 \times \mathbb{G}_2$, and computations are much faster in $\mathbb{G}_1$ than in $\mathbb{G}_2$. For the protocol of Arfaoui *et al.* we are obliged to perform exponentiations in both $\mathbb{G}_1$ and $\mathbb{G}_2$ on `MNT159`; however, for our scheme we can just use $\mathbb{G}_1$ and its more efficient exponentiations throughout the protocol. The difference in performance between the two protocols is therefore more significant than expected from the theoretical analysis.

Protocol	Precomputation	Key exchange	Verification	Trapdoor gen.	Extraction
Arfaoui *et. al.* [3]	$(2n)E$	$40E + 8P$	$(2n + 12)E + 2P$	$3E + 1P$	$(4n)E + 1P$
Our protocol	$(2n)E$	$39E$	$(2n + 12)E$	$3E$	$(4n)E$
Arfaoui *et al.* [3] (MNT159)	2.39ms	115.50ms	35.41ms	4.35ms	9.06ms
Our protocol (MNT159)	2.31ms	15.41ms	6.90ms	1.17ms	4.57ms
Our protocol (prime192v1)	1.53ms	10.85ms	4.76ms	0.78ms	2.99ms

Fig. 5. Theoretical and implementation results for our protocol vs. that of Arfaoui *et al.*. Theoretical results depend on the number of authorities (n), the number of exponentiations in a prime order group (E), and the number of pairing computations (P). Implementation results are averaged over 100 executions, for 3 authorities, using the settings `MNT159` and `prime192v`.

The pre-computation, verification, and key-extraction steps depend on the number of authorities. In Figure 5, we fixed those numbers (to n for the theoretical analysis and for 3 for the implementation analysis). In Figure 6, we evaluate the performance of these algorithms as a function of the number of authorities (ranging from 2 to 15), highlighting the linear complexity of these algorithms and the significantly-increased efficiency of our approach.

The graph in Figure 6 clearly shows that the Arfaoui *et al.* remains the least efficient one of the three (the top curve in each of the graphs). The gap between the scheme in [3] and our protocol in the `MNT159` setting shows the advantage of not using pairings (the green curve is much lower than the red curve for the verification and extraction algorithms respectively). The vertical between the two curves is much more significant in the case of verification (for which we require two pairings and several operations in $\mathbb{G}_2$ in [3]) than for extraction (one pairing only). In the case of the precomputation algorithm, neither pairings nor operations in $\mathbb{G}_2$ are needed for [3] and therefore the two curves are very close.

The `prime192v` setting is even more advantageous to our protocol, since the exponentiations are much faster, and in addition, we require no pairings. This is depicted in Figure 6, in the bottom curve (the blue one), which not only starts out being faster, but shows a much less-steep linear progression as n increases.

[7] We note that the protocol of [3] only works for a single set of authorities on both sides (Alice's a Bob's sides, respectively). In our protocol, Alice's side could be opened by a different set of authorities than Bob's side. To make the comparison fair with respect to [3], we fix the number of authorities for the Arfaoui *et al.* protocol [3] to 3, and assume for our protocol that either Alice's or Bob's side features 3 authorities, and that this is the side that opens the protocol.

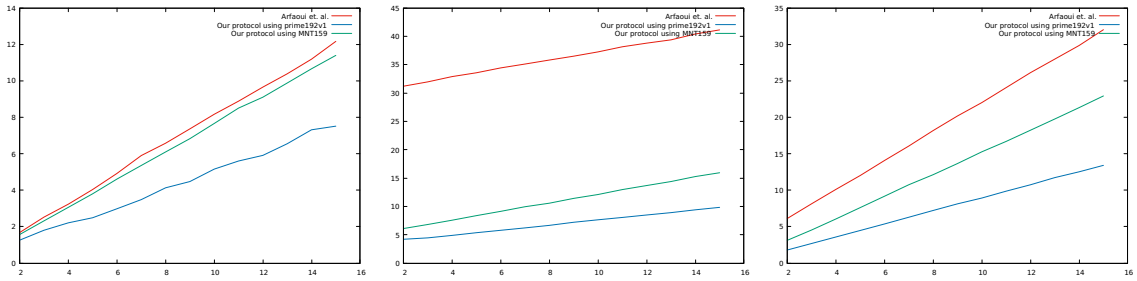

Fig. 6. Average execution time (in milliseconds, for 100 executions) of the precomputation (left), verification (middle), and extraction (right) algorithms respectively, as a function of the number of authorities. For each algorithm, we plot the graph for Arfaoui *et al.* [3] in the `MNT159` setting (red) and for our protocol using the settings `MNT159` (green) and `prime192v` (blue).

To conclude, our evaluation shows that our protocol not only improves the functionality of LIKE protocols to allow them to function in two distinct countries, but also substantially improves its effectiveness in both theory and practice.

6 Security Analysis

Our protocol guarantees the following properties, formalized by Arfaoui *et al.* for domestic mobile communications [3]:

- **Key-Security (KS):** The key computed by Alice and Bob is indistinguishable from random to anyone but Alice and Bob, as long as at least one of the authorities on Alice's side, and one of the authorities on Bob's side of the communication disagree with Lawful Interception.
- **Non-Frameability (NF):** If a user has not taken part in a particular protocol session, no one can accuse that user of having done so.
- **Honest Operator (HO):** If an operator has allowed a protocol session to run to completion, it is guaranteed that the key retrieved by lawful interception by the authorities used by the operator will yield the key extracted from Alice's and Bob's session state, *i.e.*, the key Alice and Bob themselves should have computed.

The notions our protocol achieve are slightly different than those guaranteed by the scheme of Arfaoui *et al.*, since we need to adapt the definitions to having two sets of authorities. We try to keep the modifications at a minimum. Due to page limitations, we leave the precise formalization, our proof of completeness, and our security proofs to a full version of our paper, published anonymously at [14].

7 Conclusion and future work

In this paper, we described a provably-secure, pairing-free protocol that allows Alice and Bob to communicate securely over a mobile network, without either of their serving networks learning the contents of their exchanges. In accordance with mobile standards, we allow limited, fine-grained exceptional access to the data to a group of authorities.

The main contribution of our work is that it can provide those strong properties even in the case of roaming, when Alice's communication is susceptible to being opened by a different set of authorities than Bob's. A second virtue of our scheme is that it is pairing-free, much more efficient and scalable in the number of authorities compared to prior work, and its security relies on standard assumptions.

An interesting avenue for future work is investigating how this protocol could compose with current protocols used in mobile network (such as AKA).

References

1. Harold Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield "Whit" Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller,

Bruce Schneier, Michael A. Specter, and Daniel J. Weitzner. Keys under doormats. *Communications of the ACM*, 58(10), 2015.
2. Joseph Akinyele, Christina Garman, Ian Miers, Matthew Pagano, Michael Rushanan, Matthew Green, and Aviel Rubin. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 2013.
3. Ghada Arfaoui, Olivier Blazy, Xavier Bultel, Pierre-Alain Fouque, Thibaut Jacques, Adina Nedelcu, and Cristina Onete. How to (legally) keep secrets from mobile operators. In *Proceedings of ESORICS*, LNCS. Springer, 2021.
4. Abdullah Azfar. Implementation and Performance of Threshold Cryptography for Multiple Escrow Agents in VoIP. In *Proceedings of SPIT/IPC*, pages 143–150, 2011.
5. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
6. Mihir Bellare and Shafi Goldwasser. Verifiable Partial Key Escrow. In *CCS '97*. ACM, 1997.
7. Mihir Bellare and Ronald L. Rivest. Translucent Cryptography - An Alternative to Key Escrow, and Its Implementation via Fractional Oblivious Transfer. *J. Cryptology*, 12(2), 1999.
8. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.
9. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006.
10. Liqun Chen, Dieter Gollmann, and Chris J. Mitchell. Key escrow in mutually mistrusting domains. In *Proceedings of Security Protocols*, pages 139–153, 1996.
11. M. Chen. Escrowable identity-based authenticated key agreement in the standard model. In *Chinese Electronics Journal*, volume 43, pages 1954–1962, 10 2015.
12. Dorothy E. Denning and Dennis K. Branstad. A taxonomy for key escrow encryption systems. *Commun. ACM*, 39(3), 1996.
13. Qiang Fan, Mingjian Zhang, and Yue Zhang. Key Escrow Scheme with the Cooperation Mechanism of Multiple Escrow Agents. 2012.
14. Anonymized for submission. Full version and code for our paper. `https://github.com/pairingfreelike/pairingfreelike`, 2021.
15. Yu Long, Zhenfu Cao, and Kefei Chen. A dynamic threshold commercial key escrow scheme based on conic. *Appl. Math. Comput.*, 171(2):972–982, 2005.
16. Yu Long, Kefei Chen, and Shengli Liu. Adaptive Chosen Ciphertext Secure Threshold Key Escrow Scheme from Pairing. *Informatica, Lith. Acad. Sci.*, 17(4):519–534, 2006.
17. Keith M. Martin. Increasing Efficiency of International Key Escrow in Mutually Mistrusting Domains. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 221–232. Springer, 1997.
18. Silvio Micali. Fair Public-Key Cryptosystems. In *CRYPTO '92*, volume 740 of *LNCS*. Springer, 1992.
19. Crypto Museum. Clipper chip. Available at https://www.cryptomuseum.com/crypto/usa/clipper.htm.
20. Liang Ni, Gongliang Chen, and Jianhua Li. Escrowable identity-based authenticated key agreement protocol with strong security. *Comput. Math. Appl.*, 65(9):1339–1349, 2013.
21. Adi Shamir. Partial key escrow: A new approach to software key escrow. Presented at Key Escrow Conference, 1995.
22. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53, 1984.
23. Zhen Wang, Zhaofeng Ma, Shoushan Luo, and Hongmin Gao. Key escrow protocol based on a tripartite authenticated key agreement and threshold cryptography. *IEEE Access*, 7:149080–149096, 2019.
24. Charles V. Wright and Mayank Varia. Crypto crumple zones: Enabling limited access without mass surveillance. In *Proceedings of EuroS&P 2018*. IEEE, 2018.

A Preliminaries

A.1 Notations

Let $\lambda \in \mathbb{N}$ be a security parameter. We denote by $x \leftarrow y$ the fact that a variable x is assigned a value y. We write $x \xleftarrow{\$} A$ to indicate that x is sampled identically and uniformly at random from the set A, and, for an algorithm Alg, the notation $y \leftarrow \mathsf{Alg}(x)$ expresses the fact that, if run on input x, Alg outputs y.

Many of our algorithms are probabilistic polynomial time in the (implicit) security parameter λ, a fact we denote by "PPT".

Our LIKE construction in Section 4 will require the use of digital signatures, and proofs and signatures of knowledge. Its security relies mainly on the hardness of the Decisional Diffie-Hellman problem (DDH). We review the DDH assumption and our building blocks in this section.

A.2 Assumptions and building blocks

Definition 4 (Decisional Diffie-Hellman problem). *Let $(\mathbb{G}, p, g)$ be a prime order group. Then the* decisional Diffie-Hellman problem *is* hard *if, for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$ is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda) = \left| \mathsf{Pr}\left[\begin{array}{l} (x,y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2; \\ b \leftarrow \mathcal{A}(g^x, g^y, g^{x\cdot y}) \end{array} : b = 1 \right] \right.$$
$$\left. - \mathsf{Pr}\left[\begin{array}{l} (x,y,z) \xleftarrow{\$} (\mathbb{Z}_p^*)^3; \\ b \leftarrow \mathcal{A}(g^x, g^y, g^z); \end{array} : b = 1 \right] \right|.$$

We denote by $\mathsf{Adv}^{\mathsf{DDH}}(\lambda)$ the maximum advantage a PPT adversary can have to solve this problem (note that if DDH is hard, then $\mathsf{Adv}^{\mathsf{DDH}}(\lambda)$ will be negligible as a function of the security parameter).

Definition 5 (Digital signatures). *A digital signature scheme* DS *is a triplet of algorithms* $(\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$. *The randomized key-generation* SGen *takes in input* 1^λ *and outputs a public/secret key pair* $(\mathsf{PK}, \mathsf{SK})$. *On input* SK *and a message* m*, the randomized signing algorithm* SSig *outputs a signature* σ*. The deterministic verification algorithm* SVer *takes as input the public key* PK*, the message* m*, and the signature* σ*, outputting* 1 *if the signature is deemed valid, and* 0 *otherwise. For all* $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{SGen}$ *and* m*, we assume that* $1 \leftarrow \mathsf{SVer}(\mathsf{PK}, m, \mathsf{SSig}(\mathsf{SK}, m))$ *(the signature scheme has perfect correctness).*

Let $\mathsf{DS} = (\mathsf{SGen}, \mathsf{SSig}, \mathsf{SVer})$ be a digital signature scheme. Let Sig be an oracle which, given in input the message m, internally runs $\mathsf{SSig}(\mathsf{SK}, m)$ to output a valid signature for m. The Existential Unforgeability against Chosen Message Attacks property is defined by means of the following experiment, where $\mathcal{A}$ is a PPT adversary with access to Sig:

$\mathsf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A})$:
$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{SGen}(1^\lambda)$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sig}(\cdot)}(\mathsf{PK})$
Return 1 if (m, σ) not output by $\mathsf{Sig}(\cdot)$ and $\mathsf{SVer}(\mathsf{PK}, m, \sigma) = 1$,
0 otherwise.

Definition 6 (EUF-CMA). *A digital signature* DS *is* existentially unforgeable against chosen message attacks *if, for all PPT adversaries,* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{A}}(\lambda)$ *is negligible in* λ*,*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{A}}(\lambda) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{A}}(\lambda) = 1\right].$$

We denote by $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)$ *the maximum advantage obtainable by a PPT adversary* $\mathcal{A}$*.*

Signatures of Knowledge. For our construction we require two somewhat-similar primitives: non-interactive zero-knowledge proofs of knowledge [5], and signatures of knowledge [9]. The former of these allows a prover to convince a verifier that it knows a *witness* to a particular *statement*, without revealing information about that witness. Signatures of knowledge allow a signer to bind knowledge of a witness to a statement, and a message, proving that the possessor of a valid witness to a specific statement

has signed a particular message m. We use the Camenisch/Stadler notation [8] to formalize both these primitives below.

We consider a setup in which $\mathcal{R}$ is a binary relation, and $\mathcal{L}$ is a language such that $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. We call s a statement, while w is called a witness. A *Non-Interactive Proof of Knowledge (NIPoK)* allows a prover to convince a verifier (in possession of s) that it knows a witness w such that $(s, w) \in \mathcal{R}$. In addition, a Zero-Knowledge (ZK) proof of knowledge allows no additional information about the witness to leak. We denote by $\mathsf{NIPoK}\{w : (w, s) \in \mathcal{R}\}$ a proof of knowledge of w for the statement s, given the relation $\mathcal{R}$.

In signatures of knowledge, signers use their witnesses w as private keys, while statements s become public keys. When signing, the user proves its knowledge of w, using the message m as a label to sign it. Signatures of knowledge are verifiable by any party in possession of the statement s. Such a signature is *strongly* unforgeable because the signer requires the perfect knowledge of the secret key w.

Definition 7 (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 $(\mathsf{SoK}, \mathsf{SoKver})$ such that:*

$\mathsf{SoK}_m\{w : (s, w) \in \mathcal{R}\}$**:** *outputs a signature π.*
$\mathsf{SoKver}(m, s, \pi)$**:** *outputs a bit b.*

A signature of knowledge has the following properties:

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

The definition of non-interactive zero-knowledge proofs of knowledge follows along similar lines – except NIPoKs do not require/use messages.

A.3 Security model

The execution environment. We consider an environment in which the adversary will be able to interact with honest parties via *oracles*. Each property is formalized in terms of a *security game*, played by the adversary against a *challenger*, which manipulates the honest parties.

Each party is associated with a tuple $(\mathsf{SK}, \mathsf{PK})$, namely its long-term private and public keys (these keys are generated, depending on the type of party, by one of the $\mathsf{UKeyGen}$, $\mathsf{OKeyGen}$, or $\mathsf{AKeyGen}$ algorithms). Each party is also associated with a corruption flag γ, which is raised if the adversary queries the corruption oracle.

The users and operators run the authenticated key-exchange protocol in *sessions*. At each session, each of the four parties generates a new *instance* of itself. We quantify instances collectively, denoting by π^i_{P} the i-th instance generated during the experiment, where P denotes the corresponding party.

Each instance stores a number of *attributes*, basically storing important information related to the session in which it took part, namely:

- sid: a session identifier, including a number of session-specific values that are hopefully unique to a session. User instances running the same session should have the same session identifier.
- PID: the identifiers of all the other users (*i.e.*, , parties in the set USERS) running the same session as the given instance.
- OID: the identifiers of all the other operators running the same session as the given instance.
- AID: the identifiers of all the authorities included *by that instance* in the given session. Note that, unlike for the protocol of Arfaoui *et al.*, in our scheme an instance of Alice might have a different authority partner-set than the matching instance of Bob.

– α: a flag that is raised upon a successful termination of the protocol run.
– k: the session key, which is an attribute specific only to users, and not to operators.
– sst: the session state: a set of values included in the instance's view of the session. This is an attribute specific only to operator instances.
– ρ: a reveal bit, specific only to users and set to 1 if the adversary queries the Reveal oracle on the instance's session key.
– b: a bit chosen uniformly at random upon the creation of the instance.
– τ: the transcript of the session.

For our security games, we will need to establish a correlation between session states and instances (rather than just between session states and parties, as is done in the verification algorithm). We adopt the same approach as Arfaoui *et al.* and require the existence of an auxiliary function $\mathsf{IdentifySession}(\mathsf{sst}, \pi)$, which evaluates to 1 if the given instance has run a session with state sst and 0 otherwise.

A first modification: matching conversation. We need to redefine the concept of matching instances with respect to [3], because in our case, they no longer have the same authority partner sets.

Definition 8 (Matching instances). *For any* $(i, j) \in \mathbb{N}^2$ *and* $(\mathsf{A}, \mathsf{B}) \in \mathsf{USERS}^2$ *such that* $A \neq B$*, we say that* π^i_{A} *and* π^j_{B} have matching conversation *if all the following conditions hold:* $\pi^i_{\mathsf{A}}.\mathsf{sid} \neq \perp$ *and* $\pi^i_{\mathsf{A}}.\mathsf{sid} = \pi^j_{\mathsf{B}}.\mathsf{sid}$*. If two instances* π^i_{A} *and* π^j_{B} *have matching conversation, we say that* π^i_{A} matches π^j_{B}.

Oracles. In the security games, the adversary will be able to query some or all of the oracles below (whose formal definition appears already in [3]).
– $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK}) \rightarrow \perp \cup \mathsf{P.PK}$: Creates a new party, adding it to one of the sets $\mathsf{USERS}, \mathsf{OPS}$, or AUTH, depending on the value of `role` (which can be `user`, `authority`, or `authority`. Credentials are also created, by using the appropriate Key Generation algorithm.
– $\mathsf{NewSession}(\mathsf{P}, \mathsf{PID}, \mathsf{OID}, \mathsf{AID}) \rightarrow \pi^i_{\mathsf{P}}$: Creates a new party or operator instance, with user partner set PID, operator partner set OID, and authority partner set AID.
– $\mathsf{Send}(\pi^i_{\mathsf{P}}, m) \rightarrow m'$: Sends a message m to instance π^i_{P} and returns m' according to protocol.
– $\mathsf{Reveal}(\pi^i_{\mathsf{P}}) \rightarrow \mathsf{k}$: Returns the session key (resp. the session state) if the input instance is an accepting user (resp. operator) instance, and sets that instance's reveal bit to 1.
– $\mathsf{Corrupt}(\mathsf{P}) \rightarrow \mathsf{P.SK}$: Returns a party's long-term secret key and sets that party's corruption bit to 1: $\mathsf{P}.\gamma = 1$.
– $\mathsf{Test}(\pi^i_{\mathsf{P}}) \rightarrow \widetilde{\mathsf{k}}$: This oracle can only be queried once. If the input instance belongs to a user, then, depending on the value of that instance's test bit, this oracle returns either the real key (for $\pi^i_{\mathsf{P}}.\mathsf{b} = 0$) or a random key from the same domain otherwise.
– $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda_i)_{i=1}^n, l) \rightarrow \Lambda_l.\mathsf{td}$: This oracle will reveal the trapdoor that the l-th authority in the input set of authorities would have output for a session sst if, and only if, the parameters input to this oracle verify for sst (in terms of the algorithm Verify).

Second modification: key-freshness. The notion of *key-freshness* is fundamental in typical Bellare-Rogaway models of secure authenticated key-exchange. It captures the limitations of the security guarantee that can be proved for the protocol, eliminating "trivial attacks", *i.e.*, simple attacks that the adversary can use to trivially break security. Ideally, the trivial attacks describe ways in which the protocol is *meant* to function: for instance even in a secure AKE protocol the two endpoints *must* know the session key.

The LIKE scenario defined by Arfaoui *et al.* is meant to function in a setting where only one set of authorities can exceptionally open a session key. In our case, that definition expands to two sets of authorities (the authorities on Alice's side, and those on Bob's side).

Informally, the key-freshness conditions in [3] state that the session key may only be compromised by: Alice, Bob, and the union of all the authorities for which the AKE session was run[8]. In other words, if Alice, Bob, and at least one authority remain honest and uncorrupted, then the session key is secure.

In our new definition, keys may only be compromised by: Alice, Bob, the union of all the authorities on Alice's side or the union of all the authorities on Bob's side. More formally, we define key-freshness as follows.

[8] We generalize a little here: typical AKE models also feature a key-revealing oracle, which does not necessarily map to a compromise of Alice or Bob, but just to a partial state compromise for one particular session. Key-freshness in [3] also stipulates of course that a session is no longer fresh if the key has been revealed.

Definition 9 (Key freshness). *Let π_{P}^j be the j-th instance of party $\mathsf{P} \in \mathsf{USERS}$ and let $\mathcal{A}$ be a PPT adversary against* LIKE. *Set $\mathsf{P}' \leftarrow \pi_{\mathsf{P}}^j.\mathsf{PID}$. The key $\pi_{\mathsf{P}}^j.\mathsf{k}$ is* fresh *if all the following conditions hold:*

- $\pi_{\mathsf{P}}^j.\alpha = 1$, $\mathsf{P}.\gamma = 0$ *when* $\pi_{\mathsf{P}}^j.\alpha$ *became* 1, *and* $\pi_{\mathsf{P}}^j.\rho = 0$.
- *if* π_{P}^j *matches* $\pi_{\mathsf{P}'}^k$ *for* $k \in \mathbb{N}$, *then:* $\pi_{\mathsf{P}'}^k.\alpha = 1$, $\mathsf{P}'.\gamma = 0$ *when* $\pi_{\mathsf{P}'}^k.\alpha$ *became* 1, *and* $\pi_{\mathsf{P}'}^k.\rho = 0$.
- *if no* $\pi_{\mathsf{P}'}^k$ *matches* π_{P}^j, $\mathsf{P}'.\gamma = 0$.
- $\exists\ \Lambda \in \mathsf{AUTH}$ *and there is no* $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda_i)_{i=1}^n, l)$ *query made by* $\mathcal{A}$ *such that:*
  - $\Lambda \in \pi_{\mathsf{P}}^j.\mathsf{AID}$, $\Lambda.\gamma = 0$ *and* $\Lambda = \Lambda_l$;
  - $\mathsf{IdentifySession}(\mathsf{sst}, \pi_{\mathsf{P}}^j) = 1$.
- $\exists\ \Lambda' \in \mathsf{AUTH}$ *such that for all* $k \in \mathbb{N}$ *such that* π_{P}^j *matches* $\pi_{\mathsf{P}'}^k$, *there was no* $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda_i)_{i=1}^n, l)$ *query made by* $\mathcal{A}$ *such that:*
  - $\Lambda' \in \pi_{\mathsf{P}'}^k.\mathsf{AID}$, $\Lambda'.\gamma = 0$ *and* $\Lambda' = \Lambda_l$;
  - $\mathsf{IdentifySession}(\mathsf{sst}, \pi_{\mathsf{P}'}^k) = 1$.

The security of LIKE with roaming. Since we have modified the definition of key-freshness to account for the presence of a second set of authorities, we can elegantly reuse the definition of key-security provided by Arfaoui *et al.*, as well as that for Non-Frameability (which we briefly recall below). Looking ahead, we slightly modify the definition of the key-extractor, which is simplified with respect to that of [3] – and then reuse the definition of the Honest-Operator property as is.

Key-Security. In the key-security experiment, the adversary plays the game in Figure 7 (left-hand side). It has access to all the oracles presented earlier and eventually outputs a tuple made up of an index i, a party P, and a guess bit d. The adversary wins if d is indeed the test bit associated to instance π_{P}^i (which must be an instance of P), and if that instance is fresh in the sense of the key-freshness definition. The *advantage* of $\mathcal{A}$ against $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ is defined as:

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

$\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$\mathcal{O}_{\mathsf{KS}} \leftarrow \left\{\begin{array}{l}\mathsf{Register}(\cdot,\cdot,\cdot), \mathsf{Send}(\cdot,\cdot),\\ \mathsf{NewSession}(\cdot,\cdot,\cdot),\\ \mathsf{Reveal}(\cdot), \mathsf{RevealTD}(\cdot,\cdot),\\ \mathsf{Corrupt}(\cdot,\cdot), \mathsf{Test}(\cdot)\end{array}\right\}$;
$(i, \mathsf{P}, \mathsf{d}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KS}}}(\lambda, \mathsf{pp})$;
If $\pi_{\mathsf{P}}^i.\mathsf{k}$ is fresh and $\pi_{\mathsf{P}}^i.\mathsf{b} = \mathsf{d}$, return 1;
Else $b' \xleftarrow{\$} \{0,1\}$, return b'.

$\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathcal{O}_{\mathsf{NF}} \leftarrow \left\{\begin{array}{l}\mathsf{Register}(\cdot,\cdot,\cdot), \mathsf{Send}(\cdot,\cdot), \mathsf{Reveal}(\cdot),\\ \mathsf{NewSession}(\cdot,\cdot,\cdot), \mathsf{RevealTD}(\cdot,\cdot), \mathsf{Corrupt}(\cdot,\cdot)\end{array}\right\}$;
$(\mathsf{sst}, \mathsf{P}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{NF}}}(\lambda, \mathsf{pp})$;
If $\exists\ (\mathsf{A}, \mathsf{B}) \in \mathsf{USERS}^2$, $n \in \mathbb{N}$, $\mathsf{O} \in \mathsf{OPS}$, $(\Lambda_i)_{i=1}^n \in \mathsf{AUTH}^n$ s.t.:
  $\mathsf{Verify}(\mathsf{pp}, \mathsf{sst}, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{O.PK}, (\Lambda_i.\mathsf{PK})_{i=1}^n) = 1$;
  $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$;
  $\mathsf{P}.\gamma = 0$;
  $\forall i$, if $\pi_{\mathsf{P}}^i \neq \perp$: $\mathsf{IdentifySession}(\mathsf{sst}, \pi_{\mathsf{P}}^i) = 0$ or $\pi_{\mathsf{P}}^i.\alpha = 0$,
Then return 1,
Else return 0.

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

Fig. 7. Games for key-security (KS, left), non-frameability (NF, middle), and honest-operator (HO, right).

We call the LIKE scheme *key-secure* if all PPT adversaries have at most a negligible advantage to win.

Non-Frameability. In the non-frameability game, $\mathcal{A}$ has access to all but the testing oracle, and it eventually outputs a session-state/party tuple. The adversary wins if party P never took part in (or never completed) the session that yielded sst. This is captured by the experiment in Figure 7. We define the advantage of $\mathcal{A}$ as

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

An LIKE scheme is non-frameable if all PPT adversaries have at most a negligible advantage to win the NF game.

Honest operator. In the HO game, the adversary wants to make the lawful-interception process performed on a specific completed session somehow fail (*i.e.*, retrieve a different key than should be recovered, or not to retrieve a key at all). The definition depends on a simulator called the extractor, whose job is to retrieve, from a session state, the key that Alice and Bob should have computed. Note that, since in this game Alice and Bob are malicious, they cannot be compelled to compute – or in fact use – the key that is yielded by our scheme; however, we can compel them to run the protocol correctly to completion. This is the main task of the operators, who have to check all the signatures, as well as the proofs of well-formedness of H_{A} and H_{B} provided by the endpoints.

We present here our modified key-extractor definition, which takes into account the fact that opening procedures are somewhat unilateral. In addition, we have strengthened the key-extraction requirement so that the extractor must be able to recover the session key based only on transcript equality.

Definition 10 (Key extractor). *For any* LIKE*, a key extractor* $\mathsf{Extract}(\cdot,\cdot)$ *is a deterministic unbounded algorithm such that, for any integers* n *and* m*, users* A *and* B*, operators* $\mathsf{O_A}$ *and* $\mathsf{O_B}$*, and vectors of authorities* $(\Lambda_i^{\mathsf{A}})_{i=1}^n$ *and* $(\Lambda_i^{\mathsf{B}})_{i=1}^m$*, any set* $\{\mathsf{pp}, \mathsf{A.PK}, \mathsf{A.SK}, \mathsf{B.PK}, \mathsf{B.SK}, \mathsf{O_A.PK}, \mathsf{O_A.SK}, \mathsf{O_B.PK}, \mathsf{O_B.SK}, \mathsf{k}, \mathsf{sst_A}, \mathsf{sst_B}, \mathsf{APK_A} = (\Lambda_i^{\mathsf{A}}.\mathsf{PK})_{i=1}^n, (\Lambda_i^{\mathsf{A}}.\mathsf{SK})_{i=1}^n, \mathsf{APK_B} = (\Lambda_i^{\mathsf{B}}.\mathsf{PK})_{i=1}^m, (\Lambda_i^{\mathsf{B}}.\mathsf{SK})_{i=1}^m, \tau_{\mathsf{A}}, \tau_{\mathsf{B}}, \mathsf{PPK}\}$ *generated as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$*;* $(\mathsf{A.PK}, \mathsf{A.SK}) \leftarrow \mathsf{UKeyGen}(\mathsf{pp})$*;*
$(\mathsf{B.PK}, \mathsf{B.SK}) \leftarrow \mathsf{UKeyGen}(\mathsf{pp})$*;*
$(\mathsf{O_A.PK}, \mathsf{O_A.SK}) \leftarrow \mathsf{OKeyGen}(\mathsf{pp})$*;*
$(\mathsf{O_B.PK}, \mathsf{O_B.SK}) \leftarrow \mathsf{OKeyGen}(\mathsf{pp})$*;*
$\forall i \in [\![1, n]\!]$, $(\Lambda_i^{\mathsf{A}}.\mathsf{PK}, \Lambda_i^{\mathsf{A}}.\mathsf{SK}) \leftarrow \mathsf{AKeyGen}(\mathsf{pp})$*;*
$\forall i \in [\![1, m]\!]$, $(\Lambda_i^{\mathsf{B}}.\mathsf{PK}, \Lambda_i^{\mathsf{B}}.\mathsf{SK}) \leftarrow \mathsf{AKeyGen}(\mathsf{pp})$*;*
$(\mathsf{k}, \mathsf{sst_A}, \mathsf{sst_B}, \mathsf{k}) \leftarrow \mathsf{AKE}\langle \mathsf{A}(\mathsf{A.SK}), \mathsf{O_A}(\mathsf{O_A.SK}), \mathsf{O_B}(\mathsf{O_B.SK}), \mathsf{B}(\mathsf{B.SK})\rangle(\mathsf{pp}, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{APK_A}, \mathsf{APK_B})$*;*
τ_{A} *is the transcript of the execution yielding* $\mathsf{sst_A}$ *from* $\mathsf{O_A}$*'s point of view;*
τ_{B} *is the transcript of the execution yielding* $\mathsf{sst_B}$ *from* $\mathsf{O_B}$*'s point of view;*
$\mathsf{PPK} \leftarrow \{\mathsf{O_A.PK}, \mathsf{O_B.PK}, \mathsf{A.PK}, \mathsf{B.PK}\} \cup \{\Lambda_i^{\mathsf{A}}.\mathsf{PK}\}_{i=1}^n \cup \{\Lambda_i^{\mathsf{B}}.\mathsf{PK}\}_{i=1}^m$*;*

it holds that for any $P \in \{\mathsf{A}, \mathsf{B}\}$ *and any instance* π *such that* $\pi.\tau = \tau_P$*, then:* $\mathsf{Pr}[\mathsf{Extract}(\pi, \mathsf{PPK}) = \mathsf{k}] = 1$

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).

A scheme is honest-operator secure if there exists an extractor that makes the adversary's advantage in the honest-operator game (right-hand side of Figure 7) negligible as a function of λ. The advantage is defined as: $\mathsf{Adv}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) =$

$$\mathsf{Pr}\left[\begin{array}{l|l}(k_*, \pi_{\mathsf{O}}, \mathsf{PPK}) \leftarrow \mathsf{Exp}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda); & k \neq \bot \wedge\ k_* \neq \bot \\ k \leftarrow \mathsf{Extract}(\pi_{\mathsf{O}}, \mathsf{PPK}) & \wedge\ k \neq k_*\end{array}\right].$$

A.4 Security statement

The theorem below quantifies the guarantees that our new protocol provides. We give the most complicated proof (key-security) in the appendix, but, due to space restrictions, only include sketches for the other two properties. Our full proofs can be found in our full paper [14]. Henceforth, let $\mathsf{sid} := X\|Y$, and define $\mathsf{IdentifySession}(\mathsf{sst}, \pi^j_{\mathsf{P}})$ for party P and $j \in \mathbb{N}$ as follows: parsing sst as $(b\|\mathsf{A}\|\mathsf{B}\|(\Lambda_i)_{i=1}^n\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}\|\sigma^2_{\mathsf{B}}\|\sigma_O\|H\|\mathsf{ni})$, then $\mathsf{IdentifySession}(\mathsf{sst}, \pi^j_{\mathsf{P}})$ returns 1 iff $X\|Y = \pi^i_{\mathsf{P}}.\mathsf{sid}$, and if π^j_{P} plays the role of Alice then $\mathsf{P} = \mathsf{A}$ and $\pi^j_{\mathsf{P}}.\mathsf{PID} = \mathsf{B}$, else $\pi^j_{\mathsf{P}}.\mathsf{PID} = \mathsf{A}$ and $\mathsf{P} = \mathsf{B}$.

Theorem 1. *Assuming that we instantiate our protocol with an EUF-CMA-secure signature scheme. Then our scheme:*

- *is non-frameable. Moreover, for all PPT* $\mathcal{A}$*, making at most* q_{r} *queries to* $\mathsf{Register}$*,* $\mathsf{Adv}^{\mathsf{NF}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leq q_{\mathsf{r}} \cdot \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)$.
- *is honest-operator secure if, additionally, the proofs and signatures of knowledge are zero-knowledge and extractable. Moreover, for all PPT adversaries* $\mathcal{A}$ *doing at most* q_{r} *queries to the oracle* $\mathsf{Register}$*, we have:*

$$\mathsf{Adv}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leq q_{\mathsf{r}} \cdot \leq q_{\mathsf{r}} \cdot \left(\epsilon_{\mathsf{NIPoK}}(\lambda) + \epsilon_{\mathsf{SoK}}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)\right).$$

– *is key-secure if in addition the proofs and signatures of knowledge are extractable and zero-knowledge, and the DDH assumption holds. Moreover, for all PPT adversaries $\mathcal{A}$ making at most q_{r} (resp. q_{ns}, q_{s}, and q_{t}) queries to* Register *(resp.* NewSession, Send, *and* RevealTD*):*

$$\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{s}}^2}{p} + q_{\mathsf{ns}} \cdot q_{\mathsf{r}}^2 \cdot \Big(\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda) + q_{\mathsf{ns}} \cdot q_{\mathsf{r}}^2 \cdot \Big((2 \cdot q_{\mathsf{t}} + q_{\mathsf{s}}) \cdot \epsilon_{\mathsf{SoK}}(\lambda) + q_{\mathsf{r}} \cdot \epsilon_{\mathsf{NIPoK}}(\lambda) + 3 \cdot \mathsf{Adv}^{\mathsf{DDH}}(\lambda)\Big)\Big).$$

B Correctness

Correctness of LI. Let us examine the correctness of the algorithm on one side of the conversation, say on Alice's side. The ciphertext used in the opening algorithm is $H = H_{\mathsf{A}} = (h_{\mathsf{A}} \cdot Y)^x = h_{\mathsf{A}}^x \cdot (Y^x)$. We substitute in the value of h_{A}, which yields $H = \left(\prod_{i=1}^{n} \Lambda_i^{\mathsf{A}}.\mathsf{pk}\right)^x \cdot \mathsf{k}_{\mathsf{A}} = \left(\prod_{i=1}^{n} \left(\Lambda_i^{\mathsf{A}}.\mathsf{pk}\right)^x\right) \cdot \mathsf{k}_{\mathsf{A}} = \left(\prod_{i=1}^{n} \left(g^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}\right)^x\right) \cdot \mathsf{k}_{\mathsf{A}}$. Note that we can switch the left-hand exponents, as $g^{\Lambda_i^{\mathsf{A}}.\mathsf{SK} \cdot x} = g^{x \cdot \Lambda_i^{\mathsf{A}}.\mathsf{SK}}$, and thus: $H = \left(\prod_{i=1}^{n} (g^x)^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}\right) \cdot \mathsf{k}_{\mathsf{A}} = \left(\prod_{i=1}^{n} X^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}\right) \cdot \mathsf{k}_{\mathsf{A}}$.

During Lawful Interception, each authority Λ_i^{A} generates as its first trapdoor element $\Lambda_i^{\mathsf{A}}.\mathsf{td}_1 = X^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}$ (the second element is a proof of well-formedness of that trapdoor, with respect to the authority's private key). During opening, the key is retrieved as:

$$\hat{\mathsf{k}}_{\mathsf{A}} = \frac{H_{\mathsf{A}}}{\prod_{i=1}^{n} (\Lambda_i.\mathsf{td}_1)} = \frac{H_{\mathsf{A}}}{\prod_{i=1}^{n} X^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}} = \frac{\left(\prod_{i=1}^{n} X^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}\right) \cdot \mathsf{k}_{\mathsf{A}}}{\prod_{i=1}^{n} X^{\Lambda_i^{\mathsf{A}}.\mathsf{SK}}} = \mathsf{k}_{\mathsf{A}}$$

C Security Proofs

C.1 Key-security

Proof. Let LIKE denotes our protocol. We show that $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$ is negligible for any PPT adversary $\mathcal{A}$ by using the following sequence of games:
Game G_0: This game is the same as $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$.
Game G_1: This game is similar to G_0, but aborts if the Send oracle returns twice the same element as X or Y. An abort only happens if two out of the q_{s} queried instances choose the same randomness from $\mathbb{G}$ (which is of size p), yielding:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathrm{G}_0\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathrm{G}_1\right]| \leq q_{\mathsf{s}}^2/p.$$

Let $\pi_{\mathsf{P}_*}^{i_*}$ denote a tested instance. Excluding collisions for X and Y implies that $\pi_{\mathsf{P}_*}^{i_*}$ now has at most one matching instance. Indeed, suppose two or more instances matching $\pi_{\mathsf{P}_*}^{i_*}$ exist. We parse $\pi_{\mathsf{P}_*}^{i_*}.\mathsf{sid}$ as $Z_0 \| Z_1$ where Z_i (for $i \in \{0,1\}$) was generated by $\pi_{\mathsf{P}_*}^{i_*}$.

By Def. 8, all instances matching $\pi_{\mathsf{P}_*}^{i_*}$ must sample the same $Z_{1-i} \in \mathbb{G}$ – impossible after G_1.
Game G_2: Let P_i be the i-th party instantiated by Register. Game G_2 proceeds as G_1 except that it begins by choosing $(u, v, w) \xleftarrow{\$} [\![1, q_{\mathsf{ns}}]\!] \times [\![1, q_{\mathsf{r}}]\!]^2$. If $\mathcal{A}$ returns $(i_*, \mathsf{P}_*, \mathsf{d}_*)$ such that, given $\mathsf{P}'_* \leftarrow \pi_{\mathsf{P}_*}^{i_*}.\mathsf{PID}$, we have $i_* \neq u$ or $\mathsf{P}_* \neq \mathsf{P}_v$ or $\mathsf{P}'_* \neq \mathsf{P}_w$, then G_2 aborts, returning a random bit (here, the challenger guesses the tested party instance, the associated party, and its purported partnering user). The adversary increases its winning advantage by a factor equalling the probability of guessing correctly:

$$|\mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathrm{G}_1\right] - 1/2| \leq q_{\mathsf{ns}} \cdot q_{\mathsf{r}}^2 |\mathsf{Pr}\left[\mathcal{A} \text{ wins } \mathrm{G}_2\right] - 1/2|.$$

Game G_3: Let $(i_*, \mathsf{P}_*, \mathsf{d}_*)$ be the adversary's test session that G_2 guessed. Let $\mathsf{P}'_* \leftarrow \pi^{i_*}_{\mathsf{P}_*}.\mathsf{PID}$. Game G_3 works as G_2, except that, if there exists no $\pi^k_{\mathsf{P}'_*}$ matching $\pi^{i_*}_{\mathsf{P}_*}$, the experiment aborts and returns a random bit. For any adversary $\mathcal{A}$:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_2] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_3]| \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda).$$

Assume to the contrary that there exists an adversary $\mathcal{A}$ that wins G_2 with probability $\epsilon_{\mathcal{A}}(\lambda)$ by returning a guess $(i_*, \mathsf{P}_*, \mathsf{d}_*)$ such that, setting $\mathsf{P}'_* \leftarrow \pi^{i_*}_{\mathsf{P}_*}.\mathsf{PID}$, no $k \in \mathbb{N}$ exists such that $\pi^{i_*}_{\mathsf{P}_*}$ and $\pi^k_{\mathsf{P}'_*}$ match. Game G_2 demands $\mathsf{P}'_* \leftarrow \mathsf{P}_w$ (guessed by G_2); key-freshness (Def. 9) requires P_w to be uncorrupted and ending in an accepting state. We use $\mathcal{A}$ to build a PPT adversary $\mathcal{B}$ that breaks the $\mathsf{EUF\text{-}CMA}$ security of DS with non-negligible probability. $\mathcal{B}$ receives the verification key $\hat{\mathsf{PK}}$, initializes $\mathcal{L}_S \leftarrow \emptyset$, and faithfully simulates G_2 to $\mathcal{A}$, except for $\mathcal{A}$'s following queries:

Oracle $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK})$**:** If $\mathsf{P} = \mathsf{P}_w$ with $\mathsf{P}.\mathsf{PK} = \perp$, then $\mathcal{B}$ sets $\mathsf{P}.\mathsf{PK} \leftarrow \hat{\mathsf{PK}}$.

Oracle $\mathsf{Send}(\pi^i_{\mathsf{P}}, m)$**:** There are two particular cases: $\mathsf{P} = \mathsf{P}_w$ and $\mathsf{P} = \mathsf{P}_*$. If $\mathsf{P} = \mathsf{P}_w$, then $\mathcal{B}$ queries its $\mathsf{Sig}(\cdot)$ oracle to answer $\mathcal{A}$'s queries. Depending on the role of P_w and the protocol step, $\mathcal{B}$ runs one of: $\sigma^1_{\mathsf{B}} \leftarrow \mathsf{Sig}(\mathsf{A}\|\mathsf{B}\|X\|Y)$, $\sigma_{\mathsf{A}} \leftarrow \mathsf{Sig}(\mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}})$ or $\sigma^2_{\mathsf{B}} \leftarrow \mathsf{Sig}(\mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}})$. Here, if P_w is the initiator, $\mathsf{A}\|\mathsf{B} = \mathsf{P}_w\|\mathsf{P}_w.\mathsf{PID}$; else $\mathsf{A}\|\mathsf{B} = \mathsf{P}_w.\mathsf{PID}\|\mathsf{P}_w$. The message/signature pairs are stored in $\mathcal{L}_S$. Since $\mathsf{sid} = X\|Y$, the elements X and Y, and the identities P_w and $\pi^i_{\mathsf{P}_w}.\mathsf{PID}$ are parts of the message signed in σ_{A}, σ^1_{B}, and σ^2_{B}.

If $\mathsf{P} = \mathsf{P}_*$, $i = i_*$, and $\pi^i_{\mathsf{P}}.\mathsf{PID} = \mathsf{P}_w$, if $\mathsf{SVer}(\mathsf{P}_w.\mathsf{PK}, \sigma^2_{\mathsf{B}}, \mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}) = 1$, $\mathcal{B}$ aborts, returning $(\mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}, \sigma^2_Y)$. Otherwise, if $\mathsf{SVer}(\mathsf{P}_w.\mathsf{PK}, \sigma_{\mathsf{A}}, \mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}) = 1$, $\mathcal{B}$ aborts, returning $(\mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}, \sigma_{\mathsf{A}})$.

Oracle $\mathsf{Corrupt}(\mathsf{P})$**:** If $\mathsf{P} = \mathsf{P}_w$, $\mathcal{B}$ aborts (due to G_2).

$\mathcal{B}$ wins if it sends its challenger a message/signature pair $(M, \sigma) \notin \mathcal{L}_S$ such that $\mathsf{SVer}(\hat{\mathsf{PK}}, \sigma, M) = 1$ with $\hat{\mathsf{PK}} = \mathsf{P}_w.\mathsf{PK}$. We first argue that $\mathcal{A}$ must query Send on input $\mathsf{P} = \mathsf{P}_*$, $i = i_*$, and $\pi^i_{\mathsf{P}}.\mathsf{PID} = \mathsf{P}_w$, on message $M_{\mathsf{B}} = \mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}}$ such that $\mathsf{SVer}(\mathsf{P}_w.\mathsf{PK}, \sigma^2_{\mathsf{B}}, M_{\mathsf{B}}) = 1$, or on message $M_{\mathsf{A}} = \mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}$ such that $\mathsf{SVer}(\mathsf{P}_w.\mathsf{PK}, \sigma_{\mathsf{A}}, M_{\mathsf{B}}) = 1$. Indeed, if $\mathcal{A}$ does not, the (honest) target instance $\pi^{i_*}_{\mathsf{P}_*}$ rejects.

Now we can assume that $\mathcal{A}$ has queried Send either with σ_{A} or with σ_{B} as above. We have two cases: the submitted message/signature pair is in $\mathcal{L}_S$, or it is not. If the latter happens, clearly $\mathcal{B}$ wins. Assume that the former happens, *i.e.*, the signature σ^2_{B} or σ_{A} are in $\mathcal{L}_S$ (generated by $\mathcal{B}$'s oracle). We recall that by assumption $\mathcal{A}$'s challenge instance has no matching instance, *i.e.*, there exists no $\pi^j_{\mathsf{P}_w}$ such that $\pi^j_{\mathsf{P}_w}.\mathsf{sid} \neq \perp$ or $\pi^j_{\mathsf{P}_w}.\mathsf{sid} = \pi^{i_*}_{\mathsf{P}_*}.\mathsf{sid}$. However, if $\pi^j_{\mathsf{P}_w}.\mathsf{sid} \neq \pi^{i_*}_{\mathsf{P}_*}.\mathsf{sid}$, then then $\mathcal{A}$ must have somehow completed the target session (key-freshness) and used the forged signature as input to at least one Send message (for Alice's signature or for Bob's second sig, depending on the role of P_*). This message was not created/output by $\mathcal{B}$, so it can't be in the list, so $\mathcal{B}$ also wins.

Thus, $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{B}}(\lambda) = \epsilon_{\mathcal{A}}(\lambda)$, concluding the proof.

After G_2, G_3, either a unique instance $\pi^r_{\mathsf{P}_w}$ exists, matching $\pi^u_{\mathsf{P}_v} = \pi^{i_*}_{\mathsf{P}_*}$ or the experiment returns a random bit.

Game G_4: Game G_4 runs as G_3 except that it begins by picking $r \xleftarrow{\$} [\![1, q_{\mathsf{ns}}]\!]$ (a guess for the matching instance). If $\mathcal{A}$ returns $(i_*, \mathsf{P}_*, \mathsf{d}_*)$ such that $\pi^{i_*}_{\mathsf{P}_*}$ and $\pi^r_{\mathsf{P}_w}$ do not match, then the experiment returns a random bit. The advantage of $\mathcal{A}$ on G_4 increases w.r.t. that in G_3 by a factor equalling the correct guessing probability :

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_3] - 1/2| \leq q_{\mathsf{ns}}|\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] - 1/2|.$$

Game G_5: Game G_5 proceeds as G_4, except that it begins by picking $(l_1, l_2) \xleftarrow{\$} [\![1, q_{\mathsf{r}}]\!]^2$. If the l_1-th or the l_2-th party queried to the oracle $\mathsf{Register}$ is not authority, or if it is an authority (we will denote it $\Lambda^*_{l_1}$ or $\Lambda^*_{l_2}$) who is corrupted, or if $\mathsf{RevealTD}$ is called on $(\mathsf{sst}, \mathsf{A}, \mathsf{B}, (\Lambda_i)^n_{i=1}, l)$ such that $\Lambda_l = \Lambda^*_{l_1}$ and $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) = 1$, or if $\Lambda_l = \Lambda^*_{l_2}$ and $\mathsf{IdentifySession}(\mathsf{sst}, \pi^r_{\mathsf{P}_w}.\mathsf{sid}) = 1$, then the experiment aborts by returning a random bit. Note that $\pi^u_{\mathsf{P}_v}$ is the tested instance and $\pi^r_{\mathsf{P}_w}$ is the unique instance that matches $\pi^u_{\mathsf{P}_v}$ so, by key-freshness (Def. 9), if no index l_1 and l_2 exists such that $\Lambda^*_{l_1}$ and $\Lambda^*_{l_2}$ are uncorrupted, and $\mathsf{RevealTD}$ has never been called on the query $(\mathsf{sst}, \mathsf{A}, \mathsf{B}, (\Lambda_i)^n_{i=1}, l)$ such that $\Lambda_l = \Lambda^*_{l_1}$ and $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) = 1$, and $\mathsf{RevealTD}$ has never been called on the query $(\mathsf{sst}, \mathsf{A}, \mathsf{B}, (\Lambda_i)^n_{i=1}, l)$ such that $\Lambda_l = \Lambda^*_{l_2}$ and $\mathsf{IdentifySession}(\mathsf{sst}, \pi^r_{\mathsf{P}_w}.\mathsf{sid}) = 1$, then the experiment returns a random bit. Thus, the advantage of $\mathcal{A}$ in G_5 is superior to that in G_4 by a factor equalling the guessing probability:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_4] - 1/2| \leq {q_{\mathsf{r}}}^2|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - 1/2|.$$

Game G_6: Let Ext denote the knowledge extractor of the signature of knowledge. This game is the same as G_5 except that it begins by initializing $\mathcal{L}[\,] \leftarrow \emptyset$ and:

- each time the Send oracle generates a SoK ni of an element $d \xleftarrow{\$} \mathbb{Z}_p^*$ (the exponent) for elements g_1, D_1, g_2 and D_2 such that $D_1 \leftarrow g_1^d$ and $D_2 \leftarrow g_2^d$ (D_1, D_2 have equal exponents) for the message ω, it sets $\mathcal{L}[(g_1, D_1, g_2, D_2, \omega, \mathsf{ni})] \leftarrow d$;
- each time the oracles Send or $\mathsf{RevealTD}$ verify a valid signature of knowledge $\mathsf{SoKver}(\omega, (g_1, D_1, g_2, D), \mathsf{ni}) = 1$ in a query made by $\mathcal{A}$ with $\mathcal{L}[(g_1, D_1, g_2, D_2, \omega, \mathsf{ni})] = \perp$, it runs the key extractor $\mathsf{Ext}(\lambda)$ on $\mathcal{A}$ to extract the witness d that matches the proof ni. If $g_1^d \neq D_1$ or $g_2^d \neq D_2$ then the experiment aborts by returning a random bit, else it sets $\mathcal{L}[(g_1, D_1, g_2, D_2, \omega, \mathsf{ni})] \leftarrow d$.

The difference between G_5 and G_6 is the possibility of the extractor failing when it is called. Since $\mathsf{RevealTD}$ requires 2 calls (for the verification of sst) and Send, one at each query,

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_5] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_6]| \leq (2 \cdot q_\mathsf{t} + q_\mathsf{s}) \cdot \epsilon_{\mathsf{SoK}}(\lambda).$$

From this step, for any non-simulated SoK ni (regardless of who generated it between the challenger and the adversary) on a statement (g_1, D_1, g_2, D_2) and a message ω, the list $\mathcal{L}$ stores the corresponding secret d at the index $(g_1, D_1, g_2, D_2, \omega, \mathsf{ni})$.

Game G_7: Let Ext denote the extractor of the ZK proof of knowledge $\mathsf{NIPoK}\{d : D = g^d\}$. Game G_7 runs as G_6, except it begins by initializing an empty list $\mathcal{L}'[\,] \leftarrow \emptyset$ and:

- Honest authority: if $\mathsf{Register}$ generates $(\Lambda.\mathsf{PK}, \Lambda.\mathsf{SK})$ for an authority Λ, it sets $\mathcal{L}'[\Lambda.\mathsf{PK}] \leftarrow \Lambda.\mathsf{SK}$;
- Malicious authority: if $\mathsf{Register}$ receives a query $(\Lambda, \mathtt{role}, \mathsf{PK})$ with $\mathtt{role} = \mathtt{authority}$ and $\mathsf{PK} \neq \perp$, it sets $\mathsf{PK} \leftarrow \Lambda.\mathsf{PK}$ and parses PK as $(\Lambda.\mathsf{pk}, \Lambda.\mathsf{ni})$.
  If $\mathsf{NIPoKver}((g, \Lambda.\mathsf{pk}), \Lambda.\mathsf{ni}) = 1$, G_7 runs the extractor $\mathsf{Ext}(\lambda)$ on $\mathcal{A}$ to get the witness $\Lambda.\mathsf{SK}$ for $\Lambda.\mathsf{ni}$.
  If $g^{\Lambda.\mathsf{SK}} \neq \Lambda.\mathsf{pk}$ then the experiment aborts by returning a random bit, else it sets $\mathcal{L}'[\Lambda.\mathsf{PK}] \leftarrow \Lambda.\mathsf{SK}$.

Once more, the difference between the games is the possibility that Ext fails in at least one of the calls to the registration oracle, yielding:

$$|\mathsf{Pr}[\mathcal{A} \text{ wins } G_6] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_7]| \leq q_\mathsf{r} \cdot \epsilon_{\mathsf{NIPoK}}(\lambda).$$

From this step, for any authority public key $\Lambda.\mathsf{PK}$ coupled with a non-simulated PoK ni (regardless of who generated it between the challenger and the adversary), the list $\mathcal{L}'$ stores the corresponding secret key $\Lambda.\mathsf{SK}$ at the index $\Lambda.\mathsf{PK}$.

Game G_8: This game is the same as G_7 except that during the session between $\pi^u_{\mathsf{P}_v}$ and $\pi^r_{\mathsf{P}_w}$, the group element H_{P_v} is chosen at random according to the uniform distribution on $\mathbb{G}$, and the proof $\mathsf{ni}_{\mathsf{P}_v}$ is simulated. We claim that:

$$\left|\mathsf{Pr}[\mathcal{A} \text{ wins } G_7] - \mathsf{Pr}[\mathcal{A} \text{ wins } G_8]\right| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$$

We prove this claim by reduction. We build a distinguisher $\mathcal{B}$ for a DDH instance (U_*, V_*, W_*).

In what follows, $\mathsf{Sim}_{\mathsf{NIPoK}}$ denotes the simulator of the proofs of knowledge and $\mathsf{Sim}_{\mathsf{SoK}}$, the simulator of the signature of knowledge. For the sake of simplicity, and since it is often clear from the context, we use the same notation to refer to the simulators of the two different NIPoK systems (DLog and exponent equality) that we use in our protocol.

It sets $\mathsf{pp} \leftarrow (\lambda, \mathbb{G}, p, g)$ and runs $\mathcal{A}(\mathsf{pp})$. It simulates G_7 to $\mathcal{A}$ as in the real game except for:

- $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK}) \rightarrow \mathsf{P.PK}$: On the l_1^{th} party, if $\mathtt{role} \neq \mathtt{authorities}$ or $\mathsf{PK} \neq \perp$, $\mathcal{B}$ aborts and returns a random bit, else it sets $\Lambda^*_{l_1} \leftarrow \mathsf{P}$; $\Lambda^*_{l_1}.\mathsf{pk} \leftarrow U_*$; $\Lambda^*_{l_1}.\mathsf{ni} \leftarrow \mathsf{Sim}_{\mathsf{NIPoK}}(g, U_*)$; $\Lambda^*_{l_1}.\mathsf{PK} \leftarrow (\Lambda^*_{l_1}.\mathsf{pk}, \Lambda^*_{l_1}.\mathsf{ni})$ and returns $\Lambda^*_{l_1}.\mathsf{PK}$.
- $\mathsf{Send}(\pi^i_\mathsf{P}, m)$: If $\mathsf{P} = \mathsf{P}_w$ and $i = r$, then if P plays the role of Alice, $\mathcal{B}$ sets $x_* \leftarrow x$ and $X_* \leftarrow X$, else $\mathcal{B}$ sets $y_* \leftarrow y$ and $Y_* \leftarrow Y$ (where (x, X) or (y, Y) are generated as in the real protocol). If $\mathsf{P} = \mathsf{P}_v$ and $i = u$, then:
  - if P plays the role of Alice, $\mathcal{B}$ proceeds as in G_7 except that at the first step it does not generate x and sets $X \leftarrow V_*$ and $X_* \leftarrow X$, and at the second step, it parses $\pi^u_{\mathsf{P}_v}.\mathsf{AID}$ as $(\Lambda^\mathsf{A}_j)_{j=1}^n$, it sets $\mathsf{SetAu} \leftarrow \{\Lambda^\mathsf{A}_j\}_{j=1}^n \setminus \{\Lambda^*_{l_1}\}$ and sets:
    $H_\mathsf{A} \leftarrow \prod_{\Lambda \in \mathsf{SetAu}} \left(X_*^{\mathcal{L}'[\Lambda.\mathsf{PK}]}\right) \cdot W_* \cdot X_*^{y_*}$.
    It then runs $\mathsf{ni}_\mathsf{A} \leftarrow \mathsf{Sim}_{\mathsf{SoK}}(\omega, (g, X_*, h_\mathsf{A}, H_\mathsf{A}))$, where $\omega = (\mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID})$. Finally, it sets $h_* \leftarrow h_\mathsf{A}$ and $H_* \leftarrow H_\mathsf{A}$ (where h_A is generated as in the real protocol).

- if P plays the role of Bob, then $\mathcal{B}$ proceeds as in G_7 except that it does not generate y and sets and $Y \leftarrow V_*$ and $Y_* \leftarrow Y$, then it parses $\pi^u_{\mathsf{P}_v}.\mathsf{AID}$ as $(\Lambda^{\mathsf{B}}_j)_{j=1}^m$, it sets $\mathsf{SetAu} \leftarrow \{\Lambda^{\mathsf{B}}_j\}_{j=1}^m \setminus \{\Lambda^*_{l_1}\}$ and sets:
  $H_{\mathsf{B}} \leftarrow \prod_{\Lambda \in \mathsf{SetAu}} \left(Y_*^{\mathcal{L}'[\Lambda.\mathsf{PK}]}\right) \cdot W_* \cdot Y_*^{x_*}$.
  It then runs $\mathsf{ni}_{\mathsf{B}} \leftarrow \mathsf{Sim}_{\mathsf{SoK}}(\omega, (g, Y_*, h_{\mathsf{B}}, H_{\mathsf{B}}))$, where $\omega = (\mathsf{P}_w \| \mathsf{P}_v \| \pi^u_{\mathsf{P}_v}.\mathsf{AID})$. Finally, it sets $h_* \leftarrow h_{\mathsf{B}}$ and $H_* \leftarrow H_{\mathsf{B}}$ (where h_{B} is generated as in the real protocol).

– $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, (\Lambda_i)_{i=1}^n, l)$: $\mathcal{B}$ parses sst as $(b \| \omega' \| X \| Y \| \sigma^1_{\mathsf{B}} \| \sigma_{\mathsf{A}} \| \sigma^2_{\mathsf{B}} \| \sigma_O \| H \| \mathsf{ni})$ and sets $\omega \leftarrow (A \| B \| (\Lambda_i)_{i=1}^n)$, sets $(Z_0, Z_1) \leftarrow (X, Y)$, $(Z_0^*, Z_1^*) \leftarrow (X_*, Y_*)$, and $h \leftarrow \prod_{i=1}^n \Lambda_i.\mathsf{pk}$.
  - If $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) = 1$ and $\Lambda_l = \Lambda^*_{l_1}$, then $\mathcal{B}$ aborts and returns a random bit, like in the key-freshness definition.
  - If $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) = 1$ and $\Lambda_l \neq \Lambda^*_{l_1}$, then $\mathcal{B}$ knows the secret key of Λ_l. It acts as in G_7 except that it computes $\Lambda_l.\mathsf{td}_1 \leftarrow Y^{\mathcal{L}[\Lambda_l.\mathsf{PK}]}$, $\Lambda_l.\mathsf{td}_2 \leftarrow \mathsf{Sim}_{\mathsf{NIPoK}}(g, \Lambda_l.\mathsf{pk}, Z_b, \Lambda_l.\mathsf{td}_1)$ and $\Lambda_l.\mathsf{td} \leftarrow (\Lambda_l.\mathsf{td}_1, \Lambda_l.\mathsf{td}_2)$.
  - If $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) \neq 1$ and $\Lambda_l = \Lambda^*_{l_1}$, then $\mathsf{IdentifySession}(\mathsf{sst}, \pi^u_{\mathsf{P}_v}.\mathsf{sid}) \neq 1$, which implies that $X \| Y \| \mathsf{A} \| \mathsf{B} \neq X_* \| Y_* \| \mathsf{P}_v \| \mathsf{P}_w$ (or $X_* \| Y_* \| \mathsf{P}_w \| \mathsf{P}_v$ depending on who plays the roles of Alice and Bob). $\mathcal{B}$ acts as in G_7 except that:
    * if $\mathsf{A} \| \mathsf{B} \| (\Lambda_i)_{i=1}^n \neq \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (or $\mathsf{P}_w \| \mathsf{P}_v \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$, if P_v plays the role of Bob), then the algorithm $\mathcal{B}$ computes $\Lambda^*_{l_1}.\mathsf{td}_1 \leftarrow (\Lambda^*_{l_1}.\mathsf{pk})^{\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]}$; $\Lambda.\mathsf{td}_2 \leftarrow \mathsf{Sim}_{\mathsf{NIPoK}}(g, \Lambda^*_{l_1}.\mathsf{pk}, Z_b, \Lambda^*_{l_1}.\mathsf{td}_1)$ and $\Lambda^*_{l_1}.\mathsf{td} \leftarrow (\Lambda^*_{l_1}.\mathsf{td}_1, \Lambda^*_{l_1}.\mathsf{td}_2)$. In this case, $\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]$ (recall this is the list in Game G_6) is always defined because $\omega \neq \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (or $\mathsf{P}_w \| \mathsf{P}_v \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ if P_v plays the role of Bob).
    * if $\mathsf{A} \| \mathsf{B} \| (\Lambda_i)_{i=1}^n = \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (or $\mathsf{P}_w \| \mathsf{P}_v \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$, if P_v plays the role of Bob) and $Z_{1-b} = Z^*_{1-b}$, then $Z_b \neq Z^*_b$, and the algorithm $\mathcal{B}$ computes $\Lambda^*_{l_1}.\mathsf{td}_1 \leftarrow (\Lambda^*_{l_1}.\mathsf{pk})^{\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]}$; $\Lambda.\mathsf{td}_2 \leftarrow \mathsf{Sim}_{\mathsf{NIPoK}}(g, \Lambda^*_{l_1}.\mathsf{pk}, Z_b, \Lambda^*_{l_1}.\mathsf{td}_1)$ and $\Lambda^*_{l_1}.\mathsf{td} \leftarrow (\Lambda^*_{l_1}.\mathsf{td}_1, \Lambda^*_{l_1}.\mathsf{td}_2)$. In this case, since $Z_b \neq Z^*_b$, then $\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]$ is always defined.
    * else if $\mathsf{A} \| \mathsf{B} \| (\Lambda_i)_{i=1}^n = \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (or $\mathsf{P}_w \| \mathsf{P}_v \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$, if P_v plays the role of Bob) and $Z_b = Z^*_b$, then $Z_{1-b} \neq Z^*_{1-b}$, which implies that:
      $h = \left(\prod_{i=1}^n \Lambda_i.\mathsf{pk}\right) \cdot Z_{1-b} \neq \left(\prod_{i=1}^n \Lambda_i.\mathsf{pk}\right) \cdot Z^*_{1-b} = h_*$,
      so $h \neq h_*$. $\mathcal{B}$ runs $\Lambda^*_{l_1}.\mathsf{td}_1 \leftarrow (\Lambda^*_{l_1}.\mathsf{pk})^{\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]}$; $\Lambda^*_{l_1}.\mathsf{td}_2 \leftarrow \mathsf{Sim}_{\mathsf{NIPoK}}(g, \Lambda^*_{l_1}.\mathsf{pk}, Z_b, \Lambda^*_{l_1}.\mathsf{td}_1)$ and $\Lambda^*_{l_1}.\mathsf{td} \leftarrow (\Lambda^*_{l_1}.\mathsf{td}_1, \Lambda^*_{l_1}.\mathsf{td}_2)$. In this case, since $h \neq h_*$, then $\mathcal{L}[(g, Z_b, h, H, \omega, \mathsf{ni})]$ is always defined.

At the end of the game, $\mathcal{A}$ returns a bit b_*, then if $b_* = \pi^u_{\mathsf{P}_v}.b$, then $\mathcal{B}$ returns 1, else it returns 0.

Let (u_*, v_*) be an element of $(\mathbb{Z}_p^*)^2$ such that $U_* = g^{u_*}$ and $V_* = g^{v_*}$. We set $\Lambda^*_{l_1}.\mathsf{SK} \leftarrow u_*$.

If P_v plays the role of Alice in $\pi^u_{\mathsf{P}_v}$, then we set $x_* \leftarrow v_*$. With these notations, we have $X_* = g^{x_*}$ and $\Lambda^*_{l_1}.\mathsf{pk} = g^{\Lambda^*_{l_1}.\mathsf{SK}}$. If $W_* = g^{u_* v_*}$, then $W_* = (\Lambda^*_{l_1}.\mathsf{pk})^{x_*}$ and:

$$\begin{aligned} H_* &= \prod_{\Lambda \in \mathsf{SetAu}} \left(X_*^{\mathcal{L}'[\Lambda.\mathsf{pk}]}\right) \cdot W_* \cdot X_*^{y_*} \\ &= \prod_{\Lambda \in \mathsf{SetAu}} \left(g^{\mathcal{L}'[\Lambda.\mathsf{pk}] \cdot x_*}\right) \cdot (\Lambda^*_{l_1}.\mathsf{pk})^{x_*} \cdot g^{x_* \cdot y_*} \\ &= \left(\prod_{i=1}^n \left(\Lambda_i^{\mathsf{A}}.\mathsf{pk}\right) \cdot Y_*\right)^{x_*} = (h_* \cdot Y_*)^{x_*} \end{aligned}$$

In this case, G_7 is perfectly simulated for $\mathcal{A}$, On the other hand, if W_* is a random value, then G_8 is perfectly simulated for $\mathcal{A}$. If P_v plays the role of Bob in $\pi^u_{\mathsf{P}_v}$, we can show in a symmetric way that if $W_* = g^{u_* v_*}$ then G_7 is perfectly simulated for $\mathcal{A}$, otherwise G_8 is perfectly simulated for $\mathcal{A}$. We deduce that:

– $\mathsf{Pr}[\mathcal{A} \text{ wins } \mathrm{G}_7] = \mathsf{Pr}\left[\begin{array}{l} (u_*, v_*) \xleftarrow{\$} (\mathbb{Z}_p^*)^2; \\ b \leftarrow \mathcal{B}(g^{u_*}, g^{v_*}, g^{u_* \cdot v_*}) \end{array} : b = 1\right]$ and

– $\mathsf{Pr}[\mathcal{A} \text{ wins } \mathrm{G}_8] = \mathsf{Pr}\left[\begin{array}{l} (u_*, v_*, w_*) \xleftarrow{\$} (\mathbb{Z}_p^*)^3; \\ b \leftarrow \mathcal{B}(g^{u_*}, g^{v_*}, g^{w_*}); \end{array} : b = 1\right]$,

which concludes the proof of the claim.

Game G_9: This game is the same as G_8 except that during the session between $\pi^u_{\mathsf{P}_v}$ and $\pi^r_{\mathsf{P}_w}$, the group element H_{P_w} is chosen at random in the uniform distribution on $\mathbb{G}$, and the proof $\mathsf{ni}_{\mathsf{P}_w}$ is simulated. We claim that:

$$\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_8\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right]\right| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$$

This claim can be proven in a similar way as for G_8.

Game G_{10}: This game is the same as G_9 except that the oracle Test always returns a random value. We claim that:

$$\left|\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right] - \mathsf{Pr}\left[\mathcal{A} \text{ wins } G_{10}\right]\right| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$$

We prove this claim by reduction. We build a distinguisher $\mathcal{B}$ against a DDH challenge (X_*, Y_*, Z_*).

It sets $\mathsf{pp} \leftarrow (\lambda, \mathbb{G}, p, g)$ and runs $\mathcal{A}(\mathsf{pp})$. It simulates G_8 to $\mathcal{A}$ as in the real game except for the following special cases:

- $\mathsf{Send}(\pi^i_{\mathsf{P}}, m)$: If $\mathsf{P} = \mathsf{P}_v$ and $i = u$, or if $\mathsf{P} = \mathsf{P}_w$ and $i = r$, then:
  - if P plays the role of Alice, then $\mathcal{B}$ proceeds as in G_7 except that it sets $X \leftarrow X_*$.
  - if P plays the role of Bob, then $\mathcal{B}$ proceeds as in G_7 except that it sets $Y \leftarrow Y_*$.

  At the end of the protocol, $\pi^i_{\mathsf{P}}.\mathsf{k}$ is not instantiated.
- $\mathsf{Test}(\pi^i_{\mathsf{P}})$: If $\mathsf{P} = \mathsf{P}_v$ and $i = u$, then it returns Z_*

At the end of the game, $\mathcal{A}$ returns a bit b_*, then if $b_* = \pi^u_{\mathsf{P}_v}.b$, then $\mathcal{B}$ returns 1, else it returns 0. If $Z_* = X_*^{y_*}$, then G_8 is perfectly simulated for $\mathcal{A}$. On the other hand, if Z_* is a random value, then G_9 is perfectly simulated for $\mathcal{A}$. We deduce that:

- $\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_8\right] = \mathsf{Pr}\left[\begin{array}{c}(x_*, y_*) \xleftarrow{\$} (\mathbb{Z}_p^*)^2; \\ b \leftarrow \mathcal{B}(g^{x_*}, g^{y_*}, g^{x_* \cdot y_*})\end{array} : b = 1\right]$ and
- $\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right] = \mathsf{Pr}\left[\begin{array}{l}(x_*, y_*, z_*) \xleftarrow{\$} (\mathbb{Z}_p^*)^3; \\ b \leftarrow \mathcal{B}(g^{x_*}, g^{y_*}, g^{z_*});\end{array} : b = 1\right]$,

which concludes the proof of the claim. Finally, since G_{10} do not depend on $\pi^u_{\mathsf{P}_v}.b$, we have that $\mathsf{Pr}\left[\mathcal{A} \text{ wins } G_9\right] = \frac{1}{2}$, cncluding the proof of the theorem.

C.2 Proof sketches: NF and HO

Due to space restrictions, we only provide sketch proofs for the non-frameability and honest-operator security statements.

Non-frameability. In the NF experiment, the adversary's goal is to make the verification algorithm believe that a given user, say Alice, has taken part in a session, although this is false. In our protocol, both Alice and Bob include at least one signature in the session state, and this signature is checked by the verification algorithm. Thus, except through forgeries, the adversary cannot win.

Honest operator. For the HO proof, we first have to provide a (non-polynomial) extractor for the key. This extractor recovers the key-shares X and Y and then finds, by exhaustive search, the discrete logarithm y of Y, using it to compute X^y. We then have to show that no adversary can cause authorities to recover, from a given sst that verifies and represents a completed session, a key different than that recovered by the extractor. The proof first uses the unforgeability of the operator's signature, essentially making sure that the sst was validated by an operator. Then, the security of the NIPoKs and SoKs ensure that the two endpoints, Alice and Bob, have in fact embedded the same key-shares (and the correct authority keys) in their auxiliary strings H_{A} and H_{B}. Once this is true, the correctness of LI guarantees the final result.