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

Anonymous Author(s)

ABSTRACT

Modern-day mobile communications allow users to connect from any place, at any time. Yet, the price of such easily-accessible communication is a lack of privacy. Current operators providing mobile service to users learn not only call- and SMS-metadata, but also the actual content of those exchanges; one of the reasons behind this is the Lawful-Interception requirement, demanding that serving networks be able to provide this data to authorities, upon receiving a warrant.

At ESORICS 2021, Arfaoui *et al.* pioneered a primitive called Lawful-Interception Key-Exchange, which seems to achieve the best of both worlds: (provably) privacy-enhanced communications, and a possibility of having a fine-grained, limited access to the user's data, if necessary. However Arfaoui *et al.*'s concept had two important shortcomings. First, it required the use of pairings which, while sufficiently efficient, might not always be available in the mobile setting. More importantly, Arfaoui *et al.*'s scheme was only usable in a domestic setting, where the two communicating users (Alice and Bob) were subject to the same Lawful-Interception authorities. The case for roaming was left as an open question.

In this paper we answer that open question, extending the framework of Arfaoui *et al.* to allow Alice and Bob, potentially in two different countries (and thus in the presence of two different sets of authorities), to establish a secure channel that provably guarantees the same, strong properties as the scheme presented at ESORICS 2021. Our construction is moreover pairing-free, and its security relies on standard assumptions. Our implementation results show that our scheme can be more than ten times faster than that of Arfaoui *et al.* for the most expensive operations (and twice faster for the least expensive ones). The performance gap only grows, moreover, as the number of authorities becomes larger.

CCS CONCEPTS

• **Security and privacy → Public key (asymmetric) techniques**; **Security protocols**.

KEYWORDS

Lawful interception, roaming, mobile networks, 5G

ACM Reference Format:

Anonymous Author(s). 2021. Pairing-free secure-channel establishment in mobile networks with fine-grained lawful interception. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 (which has led, *e.g.*, to the adoption of applications designed to protect privacy, like WhatsApp or Viber) is now stronger than ever. This desire was partially acknowledged and legislated through commendable regulations, such as the General Data Protection Regulation (GDPR[1] or the new electronic privacy regulation (ePR [2]) in the European Union. These regulations have revolutionized the way Web-browsing is done today, by regulating access to the personal data websites can collect about their 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. This is often done in the name of national and international security, and we are told it is to prevent horrific crimes, such as child abuse, terrorism, or organ trafficking. Consequently, 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[3], in which the FBI demanded a back door into Apple phones, allowing them access to locked devices. Apple refused this, explaining that such back doors are unconstitutional, and that such unrestrained access would set a dangerous precedent.

Yet, even the staunchest defenders of privacy, such as Abelson *et al.* [1] agree that limited and well-regulated exceptional access to data may be useful and acceptable (for instance, in order to prevent the heinous crimes cited above). This is not the case for universal back doors, which do not discriminate between subpoenaed and un-subpoenaed sensitive information. However, fine-grained, exceptional access to data can be viewed as constructive.

The notion of *key-escrow* is generic, capturing just such a fine-grained back door, targeting a key (or a set of keys). In spite of extensive research, key-escrow knows many impossibilities and

[1]See https://gdpr-info.eu/ for the full text.

[2]See https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52017PC0010.

[3]See https://epic.org/amicus/crypto/apple for a summary of that debate.

setbacks – due mostly to its wide-ranging application scenarios. For instance, it is usually hard to ensure that Alice and Bob use a particular channel for which they have 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 [2]. This scenario, which naturally obliges Alice and Bob to converse over a specific channel via their respective operators, also comes with enforced security properties (non-frameability and honest operator) and does not require the authorities to be online.

Although the LIKE protocol of [2] works for multiple operators (if Alice and Bob subscribe to two different service providers for instance), it does not scale well to multiple sets of authorities. In other words, exceptional access cannot work in Alice's country independently of Bob's country. This is problematic, since international cooperation might be complicated if not impossible in this case (*e.g.,* due to divergent laws or lack of mutual agreements).

The protocol presented in [2] achieves its security properties by using pairings, which require specialized libraries for the implementation, some of which might not always be readily available.

In this paper, we extend the LIKE scenario to a more realistic setup, which allows to capture both the domestic and roaming scenarios. We describe a protocol that is efficient, pairing-free, and which allows opening with respect to two independent set of authorities, one on Alice's side, and one on Bob's side. Our scheme is thus more efficient, while retaining the strong guarantees with respect to non-frameable, fine-grained lawful interception that [2] pioneered.

The LIKE setup. Lawful interception (LI) features prominently in 3GPP standards, being the process by which exceptional access can be granted to a specific set of *authorities*, in possession of a warrant, to mobile data or metadata pertaining to a *user*.

In current mobile architectures, if two users Alice and Bob decide to communicate by using mobile networks, the data that they share is forwarded by the operators serving Alice and Bob respectively. During LI, the accepted set of authorities query the operators for the data or metadata specified in the warrant. Unfortunately, this gives operators access to enormous amounts of potentially-sensitive data. Not only is this damaging to user privacy, but it also makes operators a target to any entity wanting to gain access to that data.

Arfaoui *et al.* present in [2] a pairing-based alternative to this approach, in which Alice and Bob establish an end-to-end-secure channel between them, which protects the confidentiality of messages even from the operators serving Alice and Bob. Exceptional Lawful Interception access is guaranteed by the bilinearity property, since Alice and Bob "embed" the product of an arbitrary (but fixed per each session) number of authority keys into the session key. During LI, each of the featured authorities generate a trapdoor to the session key; by then using bilinearity and the set of all the trapdoors, the authorities recover the key.

An obvious downside of the protocol is that both Alice and Bob must embed the same set of authorities into the session key; otherwise, the two keys would not match. In [2], the authors leave Lawful Interception 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 O_A. Bob is in country B, receiving mobile service from another operator O_B. When Alice and Bob communicate over the mobile network, the two operators forward all the data; however, we would ideally like Alice's and Bob's communication to remain secure even with respect to these two operators.

Unlike [2], 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 addition, since Lawful Interception can be implemented in many ways, countries A and B could require additional flexibility, such as:

- The number of authorities in the two different countries, A and B, could be different;
- While all authorities must contribute in order to achieve exceptional opening, the channel key output in that opening may only be divulged to *some* authorities;
- The operator is *a priori* not one of the opening authorities. Yet, under special circumstances, the operator might play a double part: a proxy in communication, but also an authority;
- Protocol participants in country A should not have to factor in the judicial system in country B (*i.e.,* authorities in country A and O_A need not be aware of the identities of authorities in country B);
- If LI is triggered in Country A, none of the parties in Country B need be aware of it.

In our protocol, users Alice and Bob (essentially) use signed Diffie-Hellman to establish a secure channel, by communicating via the two serving operators. The authority keys are no longer embedded in the session key; instead, in order to allow exceptional opening, the two endpoints each encrypt 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. Importantly, the ciphertext and proof are only received and verified by the operator providing service to that particular endpoint. Our protocol's security relies on standard assumptions, like the hardness of the Decisional Diffie-Hellman problem, the unforgeability of the signature scheme, and the security of simple proofs and signatures of knowledge.

We showcase the advantages of our protocol by a proof-of-concept implementation, the results of which are summarized in Section 5. We show that in a pairing-friendly setting (curve `MNT159`, which optimizes performance for our competitor, Arfaoui *et al.*), our protocol is still much faster. The gap in performance increases dramatically with the number of pairings required by Arfaoui *et al.* at each operation. Our key-extraction step is more than twice as fast for our protocol ([2] required a single pairing), whereas on `MNT159` our scheme's key-exchange is 8 times faster ([2] required 8 pairings).

The performance gap widens even more in a group that is not as pairing-friendly (thus harder to use by Arfaoui *et al.*), namely the `prime192v` setting. The performance gap is even greater (in our favour) for algorithms which have a runtime linear in the number of authorities. As we show in Section 5, the linear slope for increasing number of authorities is much steeper than ours in the protocol of Arfaoui *et al.*. Moreover, for our protocol, the `prime192v` setting seems to scale much better.

Other related work. Our work is most closely related to that of Arfaoui *et al.* [2]; however, as discussed above, this paper extends LIKE to handle two different sets of operators for the same handshake (a question left open by [2]). In addition, unlike the pairing-based construction of [2], we used standard primitives, which also reflects in the protocol runtime as shown by our implementation.

We also provide stronger guarantees than typical key-escrow protocols, such as [3, 5, 9–21]. For one thing, our chosen use-case (mobile communications) obliges participants to communicate by means of two proxying serving networks, allowing those operators to vouchsafe for Alice's and Bob's compliance to the protocol. In addition, our protocol can handle two independent sets of authorities for the same key, simultaneously, one performing LI on Alice's end, and the other, on Bob's. Finally we inherit some of the same properties as the LIKE scheme of Arfaoui *et al.*: we fine-grain access to a single session at a time; we have no central key-generation authority; we require no trust in authorities; and finally, we do not require those authorities to be online except at the moment of lawful interception opening.

A parallel line of research to ours [6, 22] aims to make LI possible, but computationally very expensive. Unfortunately, that approach contravenes 3GPP requirements towards lawful interception, which include a requirement of timely return of the subpoenaed content. We also note that this timeliness requirement is made of the operator: in other words, it is an incentive for operators to only adopt solutions which allow for speedy recovery of session keys.

2 PRELIMINARIES

2.1 Notations

Let $\lambda \in \mathbb{N}$ be a security parameter. We denote by $x \leftarrow y$ the fact that the variable x is assigned the 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.

2.2 Assumptions and building blocks

Definition 2.1 (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:

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

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 2.2 (Digital signatures). A digital signature scheme DS is a triplet of algorithms (SGen, SSig, SVer). The randomized key-generation SGen takes in input 1^λ and outputs a public/secret key pair (PK, 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 DS = (SGen, SSig, 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 2.3 (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,

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

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 [4], and signatures of knowledge [8]. 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 [7] 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 2.4 (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 (SoK, 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.

3 LIKE PROTOCOLS

Originally described by Arfaoui *et al.* [2], LIKE protocols are principally meant to allow two *users* Alice (denoted A) and Bob (denoted B), whose mobile communication is forwarded by *operators* $\mathrm{O_A}$ (providing mobile service to A) and $\mathrm{O_B}$ (serving B), to establish a secure channel that can be exceptionally opened by some *authorities*.

In the original setting of [2], 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 [2], 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 [2] 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 [2].

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.

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 [2] 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 [2] can remain unchanged, although in practice we must verify the session state output by $\mathrm{O_A}$ with respect to the authority set considered by A and her operator, while the session state output by $\mathrm{O_B}$ 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 A.PK denotes Alice's public key, and Λ_i.SK is the secret key of the i-th authority. The two operators are denoted $\mathrm{O_A}$ and $\mathrm{O_B}$ because they provide service to Alice and Bob respectively[4]. Parties A and $\mathrm{O_A}$ 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.

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

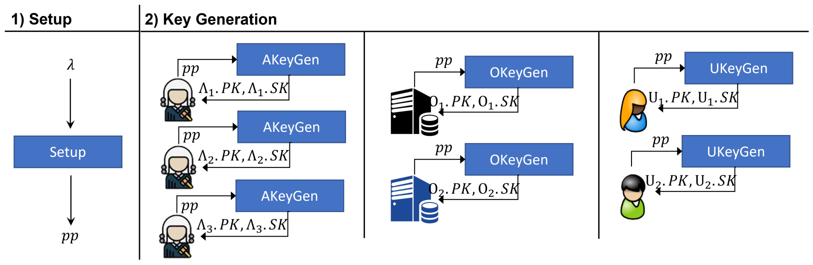


Figure 1: The Setup and Key-Generation steps, for two users, two operators, and three authorities

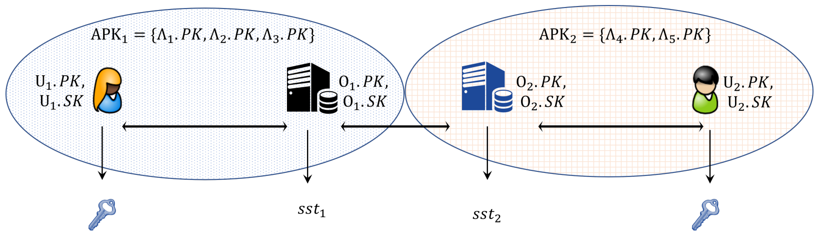


Figure 2: Authenticated key-exchange. The two left parties agree on a subset of authorities (3 in this example), while the two right parties agree on a different set of two authorities. The operators compute a session state, while the parties compute a session key.

Extending the LIKE **definition.** The only extension we make to the already-existing LIKE syntax of [2] is to provide for two sets of authorities in the AKE protocol. The new formal syntax of this algorithm is as follows:

- $\mathsf{AKE}\langle \mathsf{A}(\mathsf{A.SK}), \mathsf{O_A}(\mathsf{O_A.SK}), \mathsf{O_B}(\mathsf{O_B.SK}), \mathsf{B}(\mathsf{B.SK})\rangle(\mathsf{PK}_{\mathsf{A}\to\mathsf{B}}) \to (\mathsf{k_A}, \mathsf{sst_A}, \mathsf{sst_B}, \mathsf{k_B})$: 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 $\mathsf{PK}_{\mathsf{A}\to\mathsf{B}}$ containing: parameters pp, public keys (A.PK, 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 $\mathsf{PK}_{\mathsf{A}\to\mathsf{B}}$ 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.* [2]. 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 $\perp$.

4 OUR PROTOCOL

For our scheme, we assume that both the operators and the users employ a digital signature scheme DS = (SGen, SSig, 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.

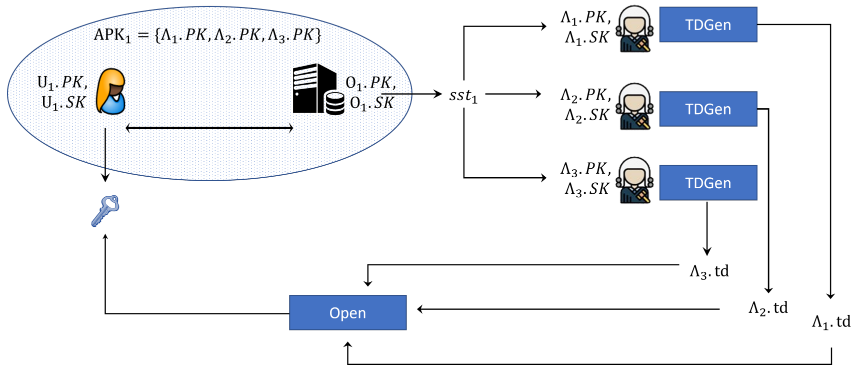

$\mathsf{APK}_1 = \{\Lambda_1.PK, \Lambda_2.PK, \Lambda_3.PK\}$
$\mathsf{U}_1.PK$, $\mathsf{U}_1.SK$
$\mathsf{O}_1.PK$, $\mathsf{O}_1.SK$
sst_1
$\Lambda_1.PK$, $\Lambda_1.SK$
$\Lambda_2.PK$, $\Lambda_2.SK$
$\Lambda_3.PK$, $\Lambda_3.SK$
TDGen
TDGen
TDGen
$\Lambda_3.\mathsf{td}$
$\Lambda_2.\mathsf{td}$
$\Lambda_1.\mathsf{td}$
Open

Figure 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.

- **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.* [2], 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 [2]. 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.

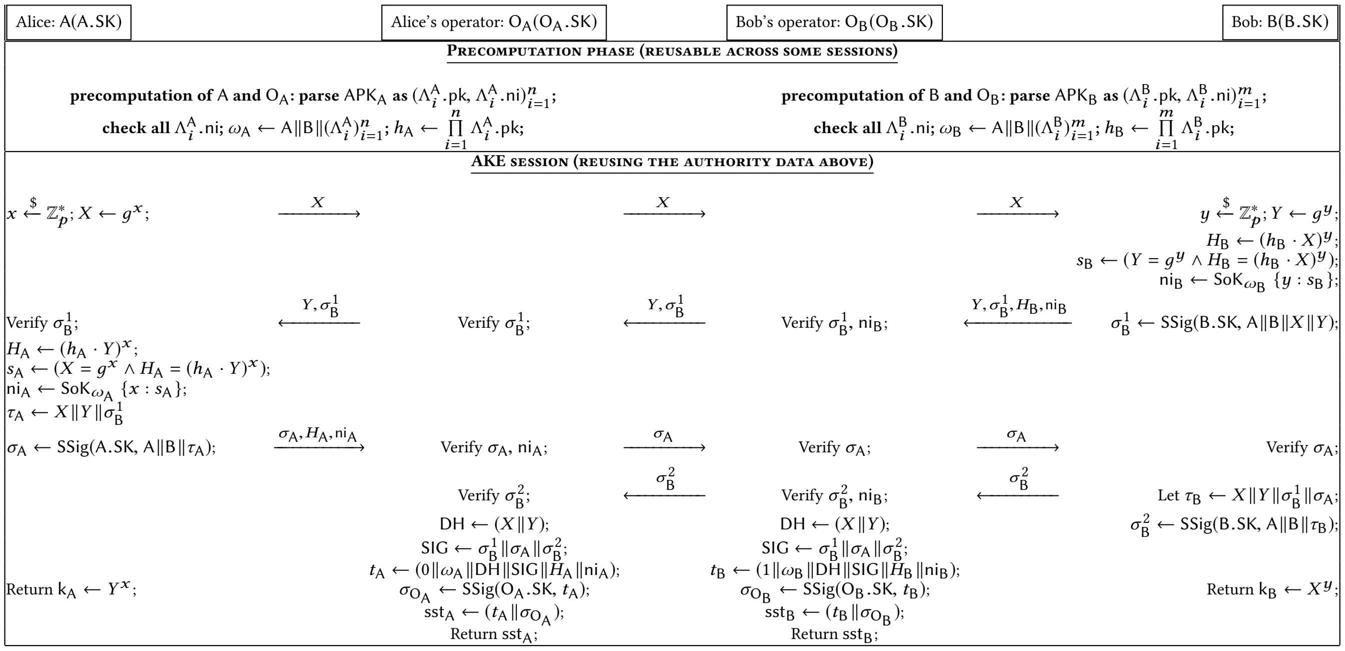
During the same precomputation phase, A and $\mathsf{O_A}$ multiply the public keys of the involved authorities, thus obtaining an auxiliary value h_A (and respectively h_B).

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 $\mathsf{O_A}$ and $\mathsf{O_B}$ is that of the signed Diffie-Hellman protocol, while the one between A and $\mathsf{O_A}$ (and B and $\mathsf{O_B}$ respectively) contains additional elements.

In order to allow the authorities in $\mathsf{APK_A}$ 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 $\mathsf{APK_B}$ 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.

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



Alice: $\mathsf{A}(\mathsf{A.SK})$		Alice's operator: $\mathsf{O_A}(\mathsf{O_A.SK})$		Bob's operator: $\mathsf{O_B}(\mathsf{O_B.SK})$		Bob: $\mathsf{B}(\mathsf{B.SK})$
Precomputation phase (reusable across some sessions)						
precomputation of A **and** $\mathsf{O_A}$**: parse** $\mathsf{APK_A}$ **as** $(\Lambda_i^{\mathsf{A}}.\mathsf{pk}, \Lambda_i^{\mathsf{A}}.\mathsf{ni})_{i=1}^n$; **check all** $\Lambda_i^{\mathsf{A}}.\mathsf{ni}$; $\omega_{\mathsf{A}} \leftarrow \mathsf{A}\Vert\mathsf{B}\Vert(\Lambda_i^{\mathsf{A}})_{i=1}^n$; $h_{\mathsf{A}} \leftarrow \prod_{i=1}^n \Lambda_i^{\mathsf{A}}.\mathsf{pk}$;				**precomputation of** B **and** $\mathsf{O_B}$**: parse** $\mathsf{APK_B}$ **as** $(\Lambda_i^{\mathsf{B}}.\mathsf{pk}, \Lambda_i^{\mathsf{B}}.\mathsf{ni})_{i=1}^m$; **check all** $\Lambda_i^{\mathsf{B}}.\mathsf{ni}$; $\omega_{\mathsf{B}} \leftarrow \mathsf{A}\Vert\mathsf{B}\Vert(\Lambda_i^{\mathsf{B}})_{i=1}^m$; $h_{\mathsf{B}} \leftarrow \prod_{i=1}^m \Lambda_i^{\mathsf{B}}.\mathsf{pk}$;		
AKE session (reusing the authority data above)						
$x \xleftarrow{\$} \mathbb{Z}_p^*; X \leftarrow g^x$;	$\xrightarrow{X}$		$\xrightarrow{X}$		$\xrightarrow{X}$	$y \xleftarrow{\$} \mathbb{Z}_p^*; Y \leftarrow g^y$; $H_{\mathsf{B}} \leftarrow (h_{\mathsf{B}} \cdot X)^y$; $s_{\mathsf{B}} \leftarrow (Y = g^y \wedge H_{\mathsf{B}} = (h_{\mathsf{B}} \cdot X)^y)$; $\mathsf{ni_B} \leftarrow \mathsf{SoK}_{\omega_{\mathsf{B}}}\{y : s_{\mathsf{B}}\}$;
Verify σ_{B}^1; $H_{\mathsf{A}} \leftarrow (h_{\mathsf{A}} \cdot Y)^x$; $s_{\mathsf{A}} \leftarrow (X = g^x \wedge H_{\mathsf{A}} = (h_{\mathsf{A}} \cdot Y)^x)$; $\mathsf{ni_A} \leftarrow \mathsf{SoK}_{\omega_{\mathsf{A}}}\{x : s_{\mathsf{A}}\}$; $\tau_{\mathsf{A}} \leftarrow X\Vert Y\Vert\sigma_{\mathsf{B}}^1$	$\xleftarrow{Y, \sigma_{\mathsf{B}}^1}$	Verify σ_{B}^1;	$\xleftarrow{Y, \sigma_{\mathsf{B}}^1}$	Verify σ_{B}^1, $\mathsf{ni_B}$;	$\xleftarrow{Y, \sigma_{\mathsf{B}}^1, H_{\mathsf{B}}, \mathsf{ni_B}}$	$\sigma_{\mathsf{B}}^1 \leftarrow \mathsf{SSig}(\mathsf{B.SK}, \mathsf{A}\Vert\mathsf{B}\Vert X\Vert Y)$;
$\sigma_{\mathsf{A}} \leftarrow \mathsf{SSig}(\mathsf{A.SK}, \mathsf{A}\Vert\mathsf{B}\Vert\tau_{\mathsf{A}})$;	$\xrightarrow{\sigma_{\mathsf{A}}, H_{\mathsf{A}}, \mathsf{ni_A}}$	Verify σ_{A}, $\mathsf{ni_A}$;	$\xrightarrow{\sigma_{\mathsf{A}}}$	Verify σ_{A};	$\xrightarrow{\sigma_{\mathsf{A}}}$	Verify σ_{A};
		Verify σ_{B}^2;	$\xleftarrow{\sigma_{\mathsf{B}}^2}$	Verify σ_{B}^2, $\mathsf{ni_B}$;	$\xleftarrow{\sigma_{\mathsf{B}}^2}$	Let $\tau_{\mathsf{B}} \leftarrow X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}$; $\sigma_{\mathsf{B}}^2 \leftarrow \mathsf{SSig}(\mathsf{B.SK}, \mathsf{A}\Vert\mathsf{B}\Vert\tau_{\mathsf{B}})$;
Return $\mathsf{k_A} \leftarrow Y^x$;		$\mathsf{DH} \leftarrow (X\Vert Y)$; $\mathsf{SIG} \leftarrow \sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2$; $t_{\mathsf{A}} \leftarrow (0\Vert\omega_{\mathsf{A}}\Vert\mathsf{DH}\Vert\mathsf{SIG}\Vert H_{\mathsf{A}}\Vert\mathsf{ni_A})$; $\sigma_{\mathsf{O_A}} \leftarrow \mathsf{SSig}(\mathsf{O_A.SK}, t_{\mathsf{A}})$; $\mathsf{sst_A} \leftarrow (t_{\mathsf{A}}\Vert\sigma_{\mathsf{O_A}})$; Return $\mathsf{sst_A}$;		$\mathsf{DH} \leftarrow (X\Vert Y)$; $\mathsf{SIG} \leftarrow \sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2$; $t_{\mathsf{B}} \leftarrow (1\Vert\omega_{\mathsf{B}}\Vert\mathsf{DH}\Vert\mathsf{SIG}\Vert H_{\mathsf{B}}\Vert\mathsf{ni_B})$; $\sigma_{\mathsf{O_B}} \leftarrow \mathsf{SSig}(\mathsf{O_B.SK}, t_{\mathsf{B}})$; $\mathsf{sst_B} \leftarrow (t_{\mathsf{B}}\Vert\sigma_{\mathsf{O_B}})$; Return $\mathsf{sst_B}$;		Return $\mathsf{k_B} \leftarrow X^y$;

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

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 $\omega \leftarrow \mathsf{A}\Vert\mathsf{B}\Vert(\Lambda_i)_{i=1}^n$ and $h \leftarrow \prod_{i=1}^n \Lambda_i.\mathsf{pk}$. Parse sst as $d\Vert\omega'\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2\Vert H\Vert\mathsf{ni}\Vert\sigma_O$ and set $Z_0 = X$ and $Z_1 = Y$. if $\omega = \omega'$ and:
  - For all $i \in [\![1, n]\!]$, $\mathsf{NIPoKver}(\Lambda_i.\mathsf{pk}, \Lambda_i.\mathsf{ni}) = 1$;
  - $\mathsf{SoKver}(\omega, (g, Z_d, h \cdot Z_{1-d}, H), \pi) = 1$;
  - $\mathsf{SVer}(\mathsf{B.PK}, \mathsf{A}\Vert\mathsf{B}\Vert X\Vert Y) = 1$;
  - $\mathsf{SVer}(\mathsf{A.PK}, \mathsf{A}\Vert\mathsf{B}\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1) = 1$;
  - $\mathsf{SVer}(\mathsf{B.PK}, \mathsf{A}\Vert\mathsf{B}\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}) = 1$;
  - $\mathsf{SVer}(\mathsf{O.PK}, d\Vert\omega'\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2\Vert H\Vert\mathsf{ni}) = 1$;

  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 $\mathsf{O_A}$. 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 $(d\Vert\omega'\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2\Vert H\Vert\mathsf{ni}\Vert\sigma_O)$ 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 $d\Vert\omega'\Vert X\Vert Y\Vert\sigma_{\mathsf{B}}^1\Vert\sigma_{\mathsf{A}}\Vert\sigma_{\mathsf{B}}^2\Vert H\Vert\mathsf{ni}\Vert\sigma_O$, 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 $\mathsf{O_A}$ outputs a session state $\mathsf{sst_A}$ 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.

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:

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

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 less 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.* [2] using the Charm-Crypto framework – which is based on Python. Since the protocol of Arfaoui *et al.* 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.* [2] 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)[5]. We evaluate the performance of our protocol for the same setting as for Arfaoui *et al.* [2] (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.

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 [2] 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 [2]) than for extraction (one pairing only). In the case of the

[5]We note that the protocol of [2] 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 [2], we fix the number of authorities for the Arfaoui *et al.* protocol [2] 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.

Protocol	Precomputation	Key exchange	Verification	Trapdoor gen.	Extraction
Arfaoui *et. al.* [2]	$(2n)E$	$40E + 8P$	$(2n + 12)E + 2P$	$3E + 1P$	$(4n)E + 1P$
Our protocol	$(2n)E$	$39E$	$(2n + 12)E$	$3E$	$(4n)E$
Arfaoui *et al.* [2] (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

Figure 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`.

precomputation algorithm, neither pairings nor operations in $\mathbb{G}_2$ are needed for [2] 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.

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 [2]:

- **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 disagrees 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. Looking ahead, we will need to change the definitions of key-freshness (for the key-security game) and of the HO extractor (for the honest-operator game). Having done so, the security experiments themselves can remain the same as in the original paper. Our new definitions, Def. 6.2 and 6.3, appear later in this section.

For completeness, we first summarize the security model due to Arfaoui *et al.*, before we introduce the theorem that quantifies the security properties of our scheme. Since the fully-formalized model exists already in [2], we prefer to use a more intuitive description of it here, only giving full formalizations for the parts that we needed to expand on or change fundamentally.

6.1 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 (SK, PK), namely its long-term private and public keys (these keys are generated, depending on the type of party, by one of the UKeyGen, OKeyGen, or 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_{P}^{i} 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.

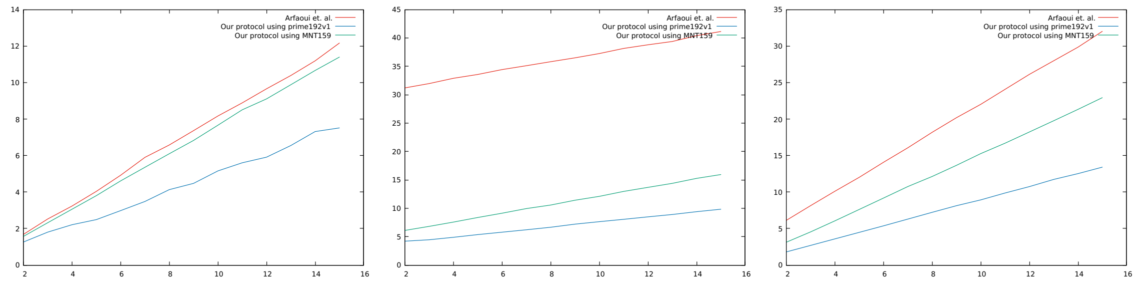


Figure 6: Average execution time (in milliseconds, for 100 executions) of the precomputation (left), verification (middle), and extraction (right) algorithms respectively, as a function of the number of authorities. For each algorithm, we plot the graph for Arfaoui *et al.* [2] in the `MNT159` setting (red) and for our protocol using the settings `MNT159` (green) and `prime192v` (blue).

- 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 IdentifySession(sst, π), 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 [2], because in our case, they no longer have the same authority partner sets.

Definition 6.1 (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 [2]).

- $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK}) \rightarrow \perp \cup \mathsf{P.PK}$: Creates a new party, adding it to one of the sets USERS, 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.
- $\mathsf{Reveal}(\pi^i_{\mathsf{P}}) \rightarrow$ 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.
- $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, \mathsf{O}, (\Lambda_i)^n_{i=1}, l) \rightarrow \Lambda_l.\mathsf{td}$: 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 [2] 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[6]. 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.

Definition 6.2 (Key freshness). Let π^j_{P} be the j-th instance of party $\mathsf{P} \in \mathsf{USERS}$ and let $\mathcal{A}$ be a PPT adversary against LIKE. Set $\mathsf{P}' \leftarrow \pi^j_{\mathsf{P}}.\mathsf{PID}$. The key $\pi^j_{\mathsf{P}}.\mathsf{k}$ is *fresh* if all the following conditions hold:

- $\pi^j_{\mathsf{P}}.\alpha = 1$, $\mathsf{P}.\gamma = 0$ when $\pi^j_{\mathsf{P}}.\alpha$ became 1, and $\pi^j_{\mathsf{P}}.\rho = 0$.
- if π^j_{P} matches $\pi^k_{\mathsf{P}'}$, for $k \in \mathbb{N}$, then: $\pi^k_{\mathsf{P}'}.\alpha = 1$, $\mathsf{P}'.\gamma = 0$ when $\pi^k_{\mathsf{P}'}.\alpha$ became 1, and $\pi^k_{\mathsf{P}'}.\rho = 0$.
- if no $\pi^k_{\mathsf{P}'}$ matches π^j_{P}, $\mathsf{P}'.\gamma = 0$.

[6]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 [2] also stipulates of course that a session is no longer fresh if the key has been revealed.

- $\exists \Lambda \in$ AUTH and there is no RevealTD(sst, A, B, O, $(\Lambda_i)_{i=1}^n, l$) query made by $\mathcal{A}$ such that:
  - $\Lambda \in \pi_{\mathsf{P}}^j$.AID, $\Lambda.\gamma = 0$ and $\Lambda = \Lambda_l$;
  - IdentifySession(sst, π_{P}^j) = 1.
- $\exists \Lambda' \in$ AUTH such that for all $k \in \mathbb{N}$ such that π_{P}^j matches $\pi_{\mathsf{P}'}^k$, there was no RevealTD(sst, A, B, O, $(\Lambda_i)_{i=1}^n, l$) query made by $\mathcal{A}$ such that:
  - $\Lambda' \in \pi_{\mathsf{P}'}^k$.AID, $\Lambda'.\gamma = 0$ and $\Lambda' = \Lambda_l$;
  - IdentifySession(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 [2] – 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 $\mathsf{Exp}_{\mathsf{LIKE},\mathcal{A}}^{\mathsf{KS}}(\lambda)$ is defined as:

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

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

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

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 6.3 (Key extractor). For any LIKE, a key extractor 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 {pp, A.PK, A.SK, B.PK, B.SK, $\mathsf{O_A}$.PK, $\mathsf{O_A}$.SK, $\mathsf{O_B}$.PK, $\mathsf{O_B}$.SK, k, $\mathsf{sst_A}$, $\mathsf{sst_B}$, $\mathsf{APK_A} = (\Lambda_i^{\mathsf{A}}.\mathsf{PK})_{i=1}^n, (\Lambda_i^{\mathsf{A}}.\mathsf{SK})_{i=1}^n$, $\mathsf{APK_B} = (\Lambda_i^{\mathsf{B}}.\mathsf{PK})_{i=1}^m, (\Lambda_i^{\mathsf{B}}.\mathsf{SK})_{i=1}^m$, $\tau_{\mathsf{A}}, \tau_{\mathsf{B}}$, PPK} generated as follows:

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

it holds that for any $P \in$ {A, B} and any instance π such that $\pi.\tau = \tau_P$, then: Pr[Extract(π, PPK) = 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: $\mathsf{Adv}_{\mathsf{LIKE},\mathcal{A}}^{\mathsf{HO}}(\lambda) =$

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

6.2 Security statement

The theorem below quantifies the guarantees that our new protocol provides. We give the proofs in the appendix. In the following, let sid $:= X\|Y$, and define IdentifySession(sst, π_{P}^j) for some party P and integer j as follows: parsing sst as $(b\|\mathsf{A}\|\mathsf{B}\|(\Lambda_i)_{i=1}^n\|X\|Y\|\sigma_{\mathsf{B}}^1\|\sigma_{\mathsf{A}}\|\sigma_{\mathsf{B}}^2\|\sigma_O\|H\|\mathsf{ni})$, the algorithm IdentifySession(sst, π_{P}^j) returns 1 iff $X\|Y = \pi_{\mathsf{P}}^i$.sid, and if π_{P}^j plays the role of Alice then P = A and π_{P}^j.PID = B, else π_{P}^j.PID = A and P = B.

Theorem 6.4. *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* Register, $\mathsf{Adv}_{\mathsf{LIKE},\mathcal{A}}^{\mathsf{NF}}(\lambda) \leq q_{\mathsf{r}} \cdot \mathsf{Adv}_{\mathsf{DS}}^{\mathsf{EUF\text{-}CMA}}(\lambda)$.
- *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* Register, *we have:*

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

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

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

Figure 7: Games for key-security (KS, left), non-frameability (NF, middle), and honest-operator (HO, 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*):*

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

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. 2015. Keys under Doormats. *Commun. ACM* 58, 10 (2015), 24âĂŞ26.

[2] Ghada Arfaoui, Olivier Blazy, Xavier Bultel, Pierre-Alain Fouque, Thibaut Jacques, Adina Nedelcu, and Cristina Onete. 2021. How to (legally) keep secrets from mobile operators. In *Proceedings of ESORICS (LNCS)*. Springer.

[3] Abdullah Azfar. 2011. Implementation and Performance of Threshold Cryptography for Multiple Escrow Agents in VoIP. In *Proceedings of SPIT/IPC*. 143–150.

[4] Mihir Bellare and Oded Goldreich. 1992. On Defining Proofs of Knowledge. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings (Lecture Notes in Computer Science, Vol. 740)*, Ernest F. Brickell (Ed.). Springer, 390–420. https://doi.org/10.1007/3-540-48071-4_28

[5] Mihir Bellare and Shafi Goldwasser. 1997. Verifiable Partial Key Escrow. In *CCS '97*. ACM.

[6] Mihir Bellare and Ronald L. Rivest. 1999. Translucent Cryptography - An Alternative to Key Escrow, and Its Implementation via Fractional Oblivious Transfer. *J. Cryptology* 12, 2 (1999).

[7] Jan Camenisch and Markus Stadler. 1997. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings (Lecture Notes in Computer Science, Vol. 1294)*, Burton S. Kaliski Jr. (Ed.). Springer, 410–424. https://doi.org/10.1007/BFb0052252

[8] Melissa Chase and Anna Lysyanskaya. 2006. On Signatures of Knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4117)*, Cynthia Dwork (Ed.). Springer, 78–96. https://doi.org/10.1007/11818175_5

[9] Liqun Chen, Dieter Gollmann, and Chris J. Mitchell. 1996. Key Escrow in Mutually Mistrusting Domains. In *Proceedings of Security Protocols*. 139–153.

[10] M. Chen. 2015. Escrowable identity-based authenticated key agreement in the standard model. In *Chinese Electronics Journal*, Vol. 43. 1954–1962.

[11] Dorothy E. Denning and Dennis K. Branstad. 1996. A Taxonomy for Key Escrow Encryption Systems. *Commun. ACM* 39, 3 (1996).

[12] Qiang Fan, Mingjian Zhang, and Yue Zhang. 2012. Key Escrow Scheme with the Cooperation Mechanism of Multiple Escrow Agents.

[13] Yu Long, Zhenfu Cao, and Kefei Chen. 2005. A dynamic threshold commercial key escrow scheme based on conic. *Appl. Math. Comput.* 171, 2 (2005), 972–982.

[14] Yu Long, Kefei Chen, and Shengli Liu. 2006. Adaptive Chosen Ciphertext Secure Threshold Key Escrow Scheme from Pairing. *Informatica, Lith. Acad. Sci.* 17, 4 (2006), 519–534.

[15] Keith M. Martin. 1997. Increasing Efficiency of International Key Escrow in Mutually Mistrusting Domains. In *Cryptography and Coding (LNCS, Vol. 1355)*. Springer, 221–232.

[16] Silvio Micali. 1992. Fair Public-Key Cryptosystems. In *CRYPTO '92 (LNCS, Vol. 740)*. Springer.

[17] Crypto Museum. [n.d.]. Clipper Chip. Available at https://www.cryptomuseum.com/crypto/usa/clipper.htm.

[18] Liang Ni, Gongliang Chen, and Jianhua Li. 2013. Escrowable identity-based authenticated key agreement protocol with strong security. *Comput. Math. Appl.* 65, 9 (2013), 1339–1349.

[19] Adi Shamir. [n.d.]. Partial key escrow: A new approach to software key escrow. Presented at Key Escrow Conference, 1995.

[20] Adi Shamir. 1984. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*. 47–53.

[21] Zhen Wang, Zhaofeng Ma, Shoushan Luo, and Hongmin Gao. 2019. Key Escrow Protocol Based on a Tripartite Authenticated Key Agreement and Threshold Cryptography. *IEEE Access* 7 (2019), 149080–149096.

[22] Charles V. Wright and Mayank Varia. 2018. Crypto Crumple Zones: Enabling Limited Access without Mass Surveillance. In *Proceedings of EuroS&P 2018*. IEEE.

A SECURITY PROOFS

A.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:

$$|\Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1]| \leq q_s^2/p.$$

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

By Def. 6.1, all instances matching $\pi^{i_*}_{\mathsf{P}_*}$ 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 $(u,v,w) \xleftarrow{\$} [\![1,q_{\mathsf{ns}}]\!] \times [\![1,q_r]\!]^2$. If $\mathcal{A}$ returns $(i_*, \mathsf{P}_*, \mathsf{d}_*)$ such that, given $\mathsf{P}'_* \leftarrow \pi^{i_*}_{\mathsf{P}_*}.\mathsf{PID}$, 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:

$$|\Pr[\mathcal{A} \text{ wins } G_1] - 1/2| \leq q_{\mathsf{ns}} \cdot q_r^2 |\Pr[\mathcal{A} \text{ wins } G_2] - 1/2|.$$

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 $\mathcal{A}$:

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

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

Oracle Send(π^i_{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 Corrupt(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 :

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

Game G_5: Game G_5 proceeds as G_4, except that it begins by picking $(l_1, l_2) \xleftarrow{\$} [\![1,q_r]\!]^2$. If the l_1-th or the l_2-th party queried to the oracle 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 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. 6.2), if no index l_1 and l_2 exists such that $\Lambda^*_{l_1}$ and $\Lambda^*_{l_2}$ are uncorrupted, and 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 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:

$$|\Pr[\mathcal{A} \text{ wins } G_4] - 1/2| \le {q_r}^2 |\Pr[\mathcal{A} \text{ wins } G_5] - 1/2|.$$

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 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 RevealTD requires 2 calls (for the verification of sst) and Send, one at each query,

$$|\Pr[\mathcal{A} \text{ wins } G_5] - \Pr[\mathcal{A} \text{ wins } G_6]| \le (2 \cdot q_t + q_s) \cdot \epsilon_{\mathsf{SoK}}(\lambda).$$

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 $\mathsf{NIPoK}\left\{d : D = g^d\right\}$. Game G_7 runs as G_6, except it begins by initializing an empty list $\mathcal{L}'[\,] \leftarrow \emptyset$ and:

- Honest authority: if 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 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:

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

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:

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

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:

- $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK}) \to \mathsf{P.PK}$: 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 $(\Lambda^{\mathsf{A}}_j)^n_{j=1}$, it sets $\mathsf{SetAu} \leftarrow \left\{\Lambda^{\mathsf{A}}_j\right\}^n_{j=1} \setminus \{\Lambda^*_{l_1}\}$ 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 $(\Lambda^{\mathsf{B}}_j)^m_{j=1}$, it sets $\mathsf{SetAu} \leftarrow \left\{\Lambda^{\mathsf{B}}_j\right\}^m_{j=1} \setminus \{\Lambda^*_{l_1}\}$ 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).
- $\mathsf{RevealTD}(\mathsf{sst}, \mathsf{A}, \mathsf{B}, (\Lambda_i)^n_{i=1}, l)$: $\mathcal{B}$ 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 $\omega \leftarrow (A \| B \| (\Lambda_i)^n_{i=1})$, sets $(Z_0, Z_1) \leftarrow (X, Y), (Z^*_0, Z^*_1) \leftarrow (X_*, Y_*)$, and $h \leftarrow \prod^n_{i=1} \Lambda_i.\mathsf{pk}$.
  - 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 $\mathsf{A} \| \mathsf{B} \| (\Lambda_i)^n_{i=1} \neq \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (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 $\mathsf{A}\|\mathsf{B}\|(\Lambda_i)^n_{i=1} = \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (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 $\mathsf{A}\|\mathsf{B}\|(\Lambda_i)^n_{i=1} = \mathsf{P}_v \| \mathsf{P}_w \| \pi^u_{\mathsf{P}_v}.\mathsf{AID}$ (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:

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

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:

- $\Pr[\mathcal{A} \text{ wins } G_7] = \Pr\left[\begin{array}{c} (u_*, v_*) \xleftarrow{\$} (\mathbb{Z}^*_p)^2; \\ b \leftarrow \mathcal{B}(g^{u_*}, g^{v_*}, g^{u_* \cdot v_*}) \end{array} : b = 1\right]$ and
- $\Pr[\mathcal{A} \text{ wins } G_8] = \Pr\left[\begin{array}{c} (u_*, v_*, w_*) \xleftarrow{\$} (\mathbb{Z}^*_p)^3; \\ b \leftarrow \mathcal{B}(g^{u_*}, g^{v_*}, g^{w_*}); \end{array} : b = 1\right]$,

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 $\mathsf{nip}_{\mathsf{P}_w}$ is simulated. We claim that:
$$\left|\Pr[\mathcal{A} \text{ wins } G_8] - \Pr[\mathcal{A} \text{ wins } G_9]\right| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$$
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:
$$\left|\Pr[\mathcal{A} \text{ wins } G_9] - \Pr[\mathcal{A} \text{ wins } G_{10}]\right| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda)$$
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:

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

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

A.2 Non-frameability

Proof. We use the following game hops:

Game G_0: The original game $\mathsf{Exp}^{\mathsf{NF}}_{\mathsf{LIKE}, \mathcal{A}}(\lambda)$.

Game G_1: Let P_i be the i-th party instantiated by Register. Game G_1 runs as G_0 except that it begins by picking $u \xleftarrow{\$} [\![1, q_r]\!]$. If $\mathcal{A}$ returns $(\mathsf{sst}, \mathsf{P})$ with $\mathsf{P} \neq \mathsf{P}_u$, then G_1 aborts and returns 0. For this game hop, we lose the guessing probability for u:
$$\Pr[\mathcal{A} \text{ wins } G_0] \leq q_r \cdot \Pr[\mathcal{A} \text{ wins } G_1].$$

We prove that $\Pr[\mathcal{A} \text{ wins } G_1] \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda)$ by reduction. Assume there exists an adversary $\mathcal{A}$ that wins G_1 with probability $\epsilon_{\mathcal{A}}(\lambda)$ by returning $(\mathsf{sst}_*, \mathsf{P}_*)$ such that $\exists\, (\mathsf{A}, \mathsf{B}) \in \mathsf{USERS}^2$, $\mathsf{O} \in \mathsf{OPS}$, $n \in \mathbb{N}$ and $(\Lambda_i)^n_{i=1} \in \mathsf{AUTH}^n$ with:

- $\mathsf{P}_* = \mathsf{P}_u$; $\mathsf{P}_* \in \{\mathsf{A}, \mathsf{B}\}$; $\mathsf{P}_*.\gamma = 0$;
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{sst}_*, \mathsf{A.PK}, \mathsf{B.PK}, \mathsf{O.PK}, (\Lambda_i.\mathsf{PK})^n_{i=1}) = 1$;
- $\forall i$, if $\pi^i_{\mathsf{P}_*} \neq \bot$ then either $\mathsf{IdentifySession}(\mathsf{sst}_*, \pi^i_{\mathsf{P}_*}) = 0$ or $\pi^i_{\mathsf{P}_*}.\alpha = 0$.

We build a PPT adversary $\mathcal{B}$ against the EUF-CMA security of DS. $\mathcal{B}$ receives the verification key $\hat{\mathsf{PK}}$, initializes a set $\mathcal{L}_S \leftarrow \emptyset$, then it simulates G_1 to $\mathcal{A}$ perfectly, except for:

– $\mathsf{Register}(\mathsf{P}, \mathtt{role}, \mathsf{PK})$**:** If $\mathsf{P} = \mathsf{P}_*$ (as guessed before) and $\mathsf{PK} = \bot$, $\mathcal{B}$ sets $\mathsf{P}_*.\mathsf{PK} \leftarrow \hat{\mathsf{PK}}$.

– $\mathsf{Send}(\pi^i_{\mathsf{P}}, m)$**:** If $\mathsf{P} = \mathsf{P}_*$, $\mathcal{B}$ simulates Send faithfully except it uses its $\mathsf{Sig}(\cdot)$ oracle for signatures. Depending on the protocol step and P_*'s role, it runs either $\sigma^1_{\mathsf{B}} \leftarrow \mathsf{Sig}(\mathsf{A}\|\mathsf{B}\|X\|Y)$, $\sigma_{\mathsf{A}} \leftarrow$

$\mathsf{Sig}(\mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}})$ or $\sigma^2_{\mathsf{B}} \leftarrow \mathsf{Sig}(\mathsf{B.SK}, \mathsf{A}\|\mathsf{B}\|X\|Y\|\sigma^1_{\mathsf{B}}\|\sigma_{\mathsf{A}})$. The message/signature pairs are stored in $\mathcal{L}_S$.
Since sid $= X\|Y$, then X, Y, $\mathsf{A} = \mathsf{P}_*$, and $\mathsf{B} = \pi^i_{\mathsf{P}_*}$.PID (or $\mathsf{A} = \pi^i_{\mathsf{P}_*}$.PID and $\mathsf{B} = \mathsf{P}_*$, depending on the party roles) are included in σ_{A}, σ^1_{B}, and σ^2_{B}.
– Corrupt(P): If $\mathsf{P} = \mathsf{P}_*$, then $\mathcal{B}$ aborts.

We parse sst$_*$ as $(b_*\|\omega_*\|X_*\|Y_*\|\sigma^1_{*,\mathsf{B}}\|\sigma_{*,\mathsf{A}}\|\sigma^2_{*,\mathsf{B}}\|\sigma_{*,O}\|H_*\|\mathsf{ni}_*)$. We denote by M_A the value $\mathsf{A}\|\mathsf{B}\|X_*\|Y_*\|\sigma^1_{*,\mathsf{B}}$, and M_B the value $\mathsf{A}\|\mathsf{B}\|X_*\|Y_*$. If Verify(pp, sst$_*$, A.PK, B.PK, O.PK, $(\Lambda_i.\mathsf{PK})^n_{i=1}) = 1$, we have that SVer(A.PK, $\sigma_{\mathsf{A}}, M_{\mathsf{A}}) = 1$ and SVer(B.PK, $\sigma^1_{\mathsf{B}}, M_{\mathsf{B}}) = 1$.

If $\forall i, \pi^i_{\mathsf{P}_*} \neq \bot$, then IdentifySession(sst$_*, \pi^i_{\mathsf{P}_*}) = 0$ or $\pi^i_{\mathsf{P}_*}.\alpha = 0$. Thus Send never outputs σ_{A} or σ^2_{B} by using the Sig(·) oracle on messages with sid $= X_*\|Y_*$, and identities A and B. We have two cases. If $\mathsf{P}_* = \mathsf{A}$, then $(M_{\mathsf{A}}, \sigma_{\mathsf{A}}) \notin \mathcal{L}_S$. If $\mathsf{P}_* = \mathsf{B}$, then $(M_{\mathsf{B}}, \sigma^2_{\mathsf{B}}) \notin \mathcal{L}_S$. Finally, if $\mathsf{P}_* = \mathsf{A}$, then $\mathcal{B}$ returns $(M_{\mathsf{A}}, \sigma_{\mathsf{A}})$. If $\mathsf{P}_* = \mathsf{B}$, then $\mathcal{B}$ returns $(M_{\mathsf{B}}, \sigma^2_{\mathsf{B}})$.

If $\mathcal{A}$ wins, then $\mathcal{B}$ returns a fresh, valid signature. Since $\mathcal{B}$'s simulation was perfect, we get $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{B}}(\lambda) = \epsilon_{\mathcal{A}}(\lambda)$, concluding the proof. □

A.3 Honest operator

Proof. We first describe the (deterministic, unbounded) key-extractor algorithm Extract(π_{O}, PPK), which extracts the two key shares X and Y from the transcript $\pi_{\mathsf{O}}.\tau$, finds the unique $y \in \mathbb{Z}^*_p$ with $g^y = Y$, and outputs $\mathsf{k} \leftarrow X^y$ (Bob's actual key).

In the HO security game, the adversary aims to output a session state sst and a series of trapdoors such that (*i*) Verify accepts the session state and (*ii*) sst can be opened with the trapdoors to return the same key given by the extractor. The adversary can corrupt all users and authorities, but not the operator. We use the following sequence of games:
Game G_0: The original game $\mathsf{Exp}^{\mathsf{HO}}_{\mathsf{LIKE},\mathcal{A}}(\lambda)$:
Game G_1: Let O_i be the i-th oracle party output by Register. Game G_1 runs as G_0 except that it begins by picking $u \xleftarrow{\$} [\![1, q_{\mathsf{r}}]\!]$. If $\mathcal{A}$ returns $(j_*, \mathsf{sst}_*, \mathsf{A}_*, \mathsf{B}_*, \mathsf{O}_*, (\Lambda_{*,i}, \Lambda_{*,i}.\mathsf{td})^n_{i=1})$ such that $\mathsf{O}_u \neq \mathsf{O}_*$, then the experiment returns 0. The game hop preserves security up to the guessing probability for operator O_*: $|\Pr[\mathcal{A}\text{ wins }\mathrm{G}_0] - 1/2| \leq q_{\mathsf{r}}|\Pr[\mathcal{A}\text{ wins }\mathrm{G}_1] - 1/2|$.
Game G_2: Let $(j_*, \mathsf{sst}_*, \mathsf{A}_*, \mathsf{B}_*, \mathsf{O}_*, (\Lambda_{*,i}, \Lambda_{*,i}.\mathsf{td})^n_{i=1})$ be $\mathcal{A}$'s guess. Game G_2 runs as G_1, except that, using $\mathsf{sid}_* = X_*\|Y_*$ where X_* and Y_* are the key shares from sst$_*$, if for all $k \in \mathbb{N}$, $\pi^k_{\mathsf{O}_*}.\mathsf{sid} \neq \mathsf{sid}_*$ or $\pi^k_{\mathsf{O}_*}.\mathsf{PID} \neq (\mathsf{A}_*, \mathsf{B}_*)$ or $\pi^k_{\mathsf{O}_*}.\mathsf{AID} \neq (\Lambda_{*,i})^n_{i=1}$, then G_2 aborts and returns 0. For any adversary $\mathcal{A}$,

$$|\Pr[\mathcal{A}\text{ wins }\mathrm{G}_0] - \Pr[\mathcal{A}\text{ wins }\mathrm{G}_1]| \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS}}(\lambda).$$

We prove this by reduction. Assume some $\mathcal{A}$ wins G_1 with probability $\epsilon_{\mathcal{A}}(\lambda)$ by returning a guess $(j_*, \mathsf{sst}_*, \mathsf{A}_*, \mathsf{B}_*, \mathsf{O}_*, (\Lambda_{*,i}, \Lambda_{*,i}.\mathsf{td})^n_{i=1})$ such that for all $k \in \mathbb{N}$, $\pi^k_{\mathsf{O}_*}.\mathsf{sid} \neq \mathsf{sid}_*$ or $\pi^k_{\mathsf{O}_*}.\mathsf{PID} \neq (\mathsf{A}_*, \mathsf{B}_*)$ or $\pi^k_{\mathsf{O}_*}.\mathsf{AID} \neq (\Lambda_{*,i})^n_{i=1}$. Note that if $\mathcal{A}$ wins G_0, then $\mathsf{O}_*.\gamma = 0$ and Verify(pp, sst$_*$, A$_*$.PK, B$_*$.PK, O$_*$.PK, $(\Lambda_{*,i}.\mathsf{PK})^n_{i=1}) = 1$.

Given the rules of G_1, if $\mathsf{O}_* \neq \mathsf{O}_u$, the game returns 0. We build a PPT adversary $\mathcal{B}$ that breaks the EUF-CMA security of DS with non-negligible advantage. $\mathcal{B}$ receives the verification key $\hat{\mathsf{PK}}$, initializes an empty set $\mathcal{L}_S \leftarrow \emptyset$, and then simulates G_0 to $\mathcal{A}$(pp) faithfully, except for:
– Register(P, `role`, PK): If the u-th register query has `role` ≠ `operator`), or PK ≠ ⊥ then $\mathcal{B}$ aborts (G_1 requirement). Else $\mathcal{B}$ sets $\mathsf{O}_u \leftarrow \mathsf{P}$ and sets $\mathsf{O}_u.\mathsf{PK} \leftarrow \hat{\mathsf{PK}}$.
– Send(π^i_{P}, m): If $\mathsf{P} = \mathsf{O}_u$, $\mathcal{B}$ simulates Send faithfully, but queries its Sig(·) oracle: $\sigma_O \leftarrow \mathsf{Sig}(t_{\mathsf{A}})$. $\mathcal{B}$ adds $(t_{\mathsf{A}}, \sigma_O)$ to $\mathcal{L}_S$ and sets sst $\leftarrow (t_{\mathsf{A}}\|\sigma_O)$.
– Corrupt(P): If $\mathsf{P} = \mathsf{O}_u$, then $\mathcal{B}$ aborts.

Finally, $\mathcal{A}$ returns $(j_*, \mathsf{sst}_*, \mathsf{A}_*, \mathsf{B}_*, \mathsf{O}_*, (\Lambda_{*,i}, \Lambda_{*,i}.\mathsf{td})^n_{i=1})$, $\mathcal{B}$ parses sst$_*$ as $t_*\|\sigma_*$, sets $\mathsf{sid}_* = X_*\|Y_*$ (X_* and Y_* are the key shares in sst$_*$), , returning $(t_*\|\sigma_*)$. Now $\mathcal{B}$ simulates G_1 perfectly for $\mathcal{A}$. If for all $k \in \mathbb{N}$, $\pi^k_{\mathsf{O}_*}.\mathsf{sid} \neq \mathsf{sid}_*$ or $\pi^k_{\mathsf{O}_*}.\mathsf{PID} \neq (\mathsf{A}_*, \mathsf{B}_*)$ or $\pi^k_{\mathsf{O}_*}.\mathsf{AID} \neq (\Lambda_{*,i})^n_{i=1}$, then sst$_*$ was not returned by Send for an operator instance, so $(t_*, \sigma_*) \notin \mathcal{L}_S$. Also, if $\mathcal{A}$ wins G_1, Verify(pp, sst$_*$, A$_*$.PK, B$_*$.PK, O$_*$.PK, $(\Lambda_{*,i}.\mathsf{PK})^n_{i=1}) = 1$, implying that SVer(O$_*$.PK, t_*) = 1, so σ_* is a valid signature on t_* for key $\hat{\mathsf{PK}}$. Moreover, $\mathsf{O}_* = \mathsf{O}_u$.

If $\mathcal{A}$ wins G_0 such that $\forall k \in \mathbb{N}$, $\pi^k_{\mathsf{O}_*}.\mathsf{sid} \neq \mathsf{sid}_*$ or $\pi^k_{\mathsf{O}_*}.\mathsf{PID} \neq (\mathsf{A}_*, \mathsf{B}_*)$ or $\pi^k_{\mathsf{O}_*}.\mathsf{AID} \neq (\Lambda_{*,i})^n_{i=1}$, then $\mathcal{B}$ wins its game and $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{B}}(\lambda) = \epsilon_{\mathcal{A}}(\lambda)$, concluding the proof.
Game G_3: Let Ext be the knowledge extractor of NIPoK $\left\{d : D_1 = g_1^d \wedge D_2 = g_2^d\right\}$. Let $(j_*, \mathsf{sst}_*, \mathsf{A}_*, \mathsf{B}_*, \mathsf{O}_*, (\Lambda_{*,i}, \Lambda_{*,i}.\mathsf{td})^n_{i=1})$ be $\mathcal{A}$'s guess. Game G_3 runs as G_2 except that:

– G_3 uses an (originally-empty) list $\mathcal{L}[\,] \leftarrow \emptyset$.
– For each RevealTD(sst, A, B, O, $(\Lambda_i)^n_{i=1}, l) \rightarrow \Lambda_l.\mathsf{td}$ query, G_3 sets $\mathcal{L}[\Lambda_l.\mathsf{td}] \leftarrow \Lambda_l.\mathsf{SK}$.
– Once $\mathcal{A}$ outputs its guess, G_3 sets $\omega_* \leftarrow \mathsf{A}_*\|\mathsf{B}_*\|(\Lambda_{*,i})^n_{i=1}$, parses state sst$_*$ as $(b_*\|\omega'_*\|X_*\|Y_*\|\sigma^1_{*,\mathsf{B}}\|\sigma_{*,\mathsf{A}}\|\sigma^2_{*,\mathsf{B}}\|\sigma_{*,O}\|H_*\|\mathsf{ni}_*)$, and sets $Z_0 \leftarrow X_*$ and $Z_1 \leftarrow Y_*$.
– For each $i \in [\![1, n]\!]$ such that $\mathcal{L}[\Lambda_{*,i}.\mathsf{td}] = \bot$, it parses $\Lambda_{*,i}$.PK as $(\Lambda_{*,i}.\mathsf{pk}, \Lambda_{*,i}.\mathsf{ni})$ and $\Lambda_{*,i}$.td as $(\Lambda_{*,i}.\mathsf{td}_1, \Lambda_{*,i}.\mathsf{td}_2)$. If 1 ← NIPoKver(($g, \Lambda_{*,i}.\mathsf{pk}, Z_{b_*}, \Lambda_{*,i}.\mathsf{td}_1), \Lambda_{*,i}.\mathsf{td}_2$), then it runs Ext($\lambda$) on $\mathcal{A}$ to extract witness w for the proof $\Lambda_{*,i}.\mathsf{td}_2$ and sets $\mathcal{L}[\Lambda_{*,i}.\mathsf{td}] \leftarrow w$. Else, G_3 aborts and returns 0. If $g^{\mathcal{L}[\Lambda_{*,i}.\mathsf{td}]} \neq \Lambda_{*,i}.\mathsf{pk}$ or $Z_{b_*}^{\mathcal{L}[\Lambda_{*,i}.\mathsf{td}]} \neq \Lambda_{*,i}.\mathsf{td}_1$, then G_3 aborts and returns 0.

G_2 and G_3 differ only if Ext fails for a valid NIPoK generated by $\mathcal{A}$, in any of the (at most) q_{r} calls. Hence:

$$|\Pr[\mathcal{A}\text{ wins }\mathrm{G}_1] - \Pr[\mathcal{A}\text{ wins }\mathrm{G}_2]| \leq q_{\mathsf{r}} \cdot \epsilon_{\mathsf{NIPoK}}(\lambda).$$

Game G_4: Let Ext be the knowledge extractor of the signature of knowledge. G_4 is like G_3 except that it uses an (initially empty) list $\mathcal{L}'[\,]$ and:
– whenever Send generates an SoK $\mathsf{ni} \leftarrow \mathsf{SoK}_{\omega}\{z : Z = g^z \wedge H = h^z\}$, it sets $\mathcal{L}'[(g, Z, h, H, \omega, \mathsf{ni})] \leftarrow z$.
– After $\mathcal{A}$'s guess, G_4 sets $\omega_* \leftarrow \mathsf{A}_*\|\mathsf{B}_*\|(\Lambda_{*,i})^n_{i=1}$, parses $\Lambda_{*,i}$.PK as $(\Lambda_{*,i}.\mathsf{pk}, \Lambda_{*,i}.\mathsf{ni})$; $\Lambda_{*,i}$.td as $(\Lambda_{*,i}.\mathsf{td}_1, \Lambda_{*,i}.\mathsf{td}_2)$'; and sst$_*$ as $(b_*\|\omega'_*\| X_*\|Y_*\|\sigma^1_{*,\mathsf{B}}\|\sigma_{*,\mathsf{A}}\|\sigma^2_{*,\mathsf{B}}\|\sigma_{*,O}\|H_*\|\mathsf{ni}_*)$. It sets $Z_0 \leftarrow X_*$, $Z_1 \leftarrow Y_*$, and $h_* \leftarrow \left(\prod_{i=1}^{n} \Lambda_{*,i}.\mathsf{pk}\right)$. If $\mathcal{L}'[(g, Z_{b_*}, h_* \cdot Z_{1-b_*}, H_*, \omega_*, \mathsf{ni}_*)] = \bot$:

- If 1 = SoKver($\omega_*, (g, Z_{b_*}, h_* \cdot Z_{1-b_*}, H_*), \mathsf{ni}_*$), G_4 runs Ext(λ) on $\mathcal{A}$, extracting witness w for proof ni_* and sets $\mathcal{L}'[(g, Z_{b_*}, h_* \cdot Z_{1-b_*}, H_*, \omega_*, \mathsf{ni}_*)] \leftarrow w$. Else, it aborts, returning 0 (note that Verify fails on sst$_*$ in this case).

- If $H_* \neq (h_* \cdot Z_{1-b_*})^{\mathcal{L}'[(g, Z_{b_*}, h_* \cdot Z_{1-b_*}, H_*, \omega_*, \mathsf{ni}_*)]}$ or if $Z_{b_*} \neq g^{\mathcal{L}'[(g, Z_{b_*}, h_* \cdot Z_{1-b_*}, H_*, \omega_*, \mathsf{ni}_*)]}$ (*i.e.*, extraction fails), then G_4 by returning 0.

If the SoK was generated by $\mathcal{A}$, then G_4 aborts with a probability equal to that of Ext failing:

$$\left| \Pr[\mathcal{A} \text{ wins } \mathrm{G}_3] - \Pr[\mathcal{A} \text{ wins } \mathrm{G}_4] \right| \leq \epsilon_{\mathsf{SoK}}(\lambda)$$

Finally, we show that $\Pr[\mathcal{A} \text{ wins } \mathrm{G}_4] = 0$. Assume an adversary $\mathcal{A}$ wins G_4 with non-zero probability. We parse $\pi_{\mathsf{O}_*}^{j_*}.\tau$ as $(X \| Y \| \sigma_{\mathsf{B}}^1 \| \sigma_{\mathsf{A}} \| \sigma_{\mathsf{B}}^2 \| \sigma_{\mathsf{O}} \| H \| \mathsf{ni})$, $\pi_{\mathsf{O}_*}^{j_*}.\mathsf{PID}$ as (A, B) and $\pi_{\mathsf{O}_*}^{j_*}.\mathsf{AID}$ as $(\Lambda_i)_{i=1}^n$. On the other hand, we parse sst_* as $(b_* \| A_* \| B_* \| (\Lambda_{*,i})_{i=1}^{n_*} X_* \| Y_* \| \sigma_{*,\mathsf{B}}^1 \| \sigma_{*,\mathsf{A}} \| \sigma_{*,\mathsf{B}}^2 \| \sigma_{*,O} \| H_* \| \mathsf{ni}_*)$. According to game G_2, $\pi_{\mathsf{O}_*}^{j_*}.\mathsf{sid} = X_* \| Y_*$ and $\pi_{\mathsf{O}_*}^{j_*}.\mathsf{PID} = (\mathsf{A}_*, \mathsf{B}_*)$ and $\pi_{\mathsf{O}_*}^{j_*}.\mathsf{AID} = (\Lambda_{*,i})_{i=1}^{n_*}$, which implies that: $X_* = X$; $Y_* = Y$; $A_* = A$; $B_* = B$; and $(\Lambda_i)_{i=1}^n = (\Lambda_{*,i})_{i=1}^{n_*}$. By definition, Extract returns $\mathsf{k} = X^y$ where $Y = g^y$. We set $h \leftarrow \left(\prod_{i=1}^n \Lambda_i.\mathsf{pk} \right)$ and $\omega = A \| B \| (\Lambda_i)_{i=1}^n$. We set z_0 and z_1 such that $g^{z_0} = Z_0$ and $g^{z_1} = Z_1$.
Algorithm Open$(\mathsf{pp}, \mathsf{sst}_*, (\Lambda_{*,i}.\mathsf{td})_{i=1}^{n_*}, (\Lambda_{*,i}.\mathsf{PK})_{i=1}^{n_*})$ returns:

$$\begin{aligned}
\mathsf{k}_* = \frac{H_*}{\prod_{i=1}^n \Lambda_i.\mathsf{td}_1} &= \frac{(h \cdot Z_{1-b_*})^{\mathcal{L}'[(g, Z_{b_*}, h \cdot Z_{1-b_*}, H_*, \omega, \mathsf{ni}_*)]}}{\prod_{i=1}^n Z_{b_*}^{\mathcal{L}[\Lambda_i.\mathsf{td}]}} \\
&= \frac{h^{\mathcal{L}'[(g, Z_{b_*}, h \cdot Z_{1-b_*}, H_*, \omega, \mathsf{ni}_*)]} \cdot g^{z_{1-b_*} \cdot \mathcal{L}'[(g, Z_{b_*}, h \cdot Z_{1-b_*}, H_*, \omega, \mathsf{ni}_*)]}}{\prod_{i=1}^n g^{z_{b_*} \cdot \mathcal{L}[\Lambda_i.\mathsf{td}]}} \\
&= \frac{\left(\prod_{i=1}^n (\Lambda_i.\mathsf{pk})^{\mathcal{L}'[(g, Z_{b_*}, h \cdot Z_{1-b_*}, H_*, \omega, \mathsf{ni}_*)]} \right) \cdot Z_{b_*}^{z_{1-b_*}}}{\prod_{i=1}^n (\Lambda_i.\mathsf{pk})^{z_{b_*}}} \\
&= \frac{\left(\prod_{i=1}^n g^{(\Lambda_i.\mathsf{SK}) \cdot \mathcal{L}'[(g, Z_{b_*}, h \cdot Z_{1-b_*}, H_*, \omega, \mathsf{ni}_*)]} \right) \cdot \mathsf{k}}{\prod_{i=1}^n (\Lambda_i.\mathsf{pk})^{z_{b_*}}} = \frac{\left(\prod_{i=1}^n Z_{b_*}^{\Lambda_i.\mathsf{SK}} \right) \cdot \mathsf{k}}{\left(\prod_{i=1}^n \Lambda_i.\mathsf{pk}^{z_{b_*}} \right)} = \mathsf{k}
\end{aligned}$$

which implies that $\mathsf{k}_* = \mathsf{k}$ with probability 1, so $\mathcal{A}$ always fails. This concludes the proof. □