Real World Deniability in Messaging
(Preliminary Version)*

Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan

EPFL, Lausanne, Switzerland
{firstname.lastname}@epfl.ch

Abstract

This work discusses real world deniability in messaging. We highlight how the different models for cryptographic deniability do not ensure practical deniability. To overcome this situation, we propose a model for real world deniability that takes into account the entire messaging system. We then discuss how deniability is (not) used in practice and the challenges arising from the design of a deniable system. We propose a simple, yet powerful solution for deniability: applications should enable direct modification of local messages; we discuss the impacts of this strong deniability property.

1 Introduction

The online dictionary `www.dictionary.com` defines *deniability* as "the ability to deny something, as knowledge of or connection with an illegal activity". Despite the negative connotation, this definition captures the coarse notion on which cryptographers seem to agree on: deniability enables a user to *plausibly deny* their participation in executing some scheme or protocol. The Signal protocol [MP16, PM16], the de-facto secure messaging standard, claims to offer deniability. One of its authors, Moxie Marlinspike, discusses deniability in the context of Off-the-Record (OTR) [BGB04] as follows [Mar13]:

> "If someone receives an OTR message from you, they can be absolutely sure you sent it (rather than having been forged by some third party), but can't prove to anyone else that it was a message you wrote."

In the cryptographic literature, deniability is generally formalised in a game played between abstract entities. A protocol is considered to be deniable if a *judge* is unable to differentiate between a real and an efficiently simulated execution of the protocol. This implies that any real execution of the protocol could have been faked. For secure messaging, a judge might be tasked to distinguish between the cryptographic conversation history on the network—that is, a sequence of ciphertexts—and a simulated transcript [RGK06, BFG+22, RG05]. Depending on the definition, the judge may be able to collude actively with one or more parties before the real transcript has been completely generated (online deniability [DKSW09, UG18]) and/or is given access to the secrets of one or more parties [BFG+22, CHMR23].

In practice, however, one must ask if these models actually (1) are accurate, and (2) can be used to actually deny something in the real world, e.g., a court of law [Gre14]. We claim

*This is an extended abstract corresponding to a talk given at RWC 2023.

in the context of secure messaging that neither of these are true. For (1), existing models fail to account for the higher-level context in which parties execute the protocol (e.g., a client that keeps messages in the device's memory and authenticates to a server), thereby rendering their deniability claims vacuous. For (2), the strong notions of deniability do not reflect how a human judge would interpret evidence in practice, both in the offline and online settings. We emphasise that we are not calling into question the *technical* value of prior work (i.e., we have no inclination that any existing results on deniability are formally incorrect), but rather their shortcomings when applied in the real world.

Motivated by these shortcomings, we propose a new model for deniability in secure messaging. Our model captures the fact that in practice, messages are routed between users via a server that in general *authenticates* users. Fix two parties, Alice and Bob, and assume that Bob incriminates Alice. In our model, the judge is given the state of Bob (the phone, for example) after allegedly communicating with Alice. The judge is also given information from the server and any other prior information she might have. The system is considered deniable if there exists a simulator who, *under application-specific constraints*, can interact with the server and produce a state that is indistinguishable from Bob's. This approach extends the classical notions that only consider the cryptographic transcript: we give the judge Bob's state, which can be the entire phone, and a portion of the server's state to make the decision alongside arbitrary auxiliary data [RGK06]. Our model therefore enlarges the purely cryptographic model by taking into account real-world evidence that can break deniability.

We argue that Signal is deniable under our notion if the application allows users to *modify messages stored on their devices*, which enables *all* users to simulate the conversation *in practice*. This approach provides concrete guarantees since, in practice, screenshots or compromised phones are generally considered authentic, and is likely much more tangible to a judge than an abstract simulator. We therefore advocate to implement message modification—the ability for parties to insert, remove and edit messages—on the local device, into Signal and other secure messaging applications. We nevertheless caution that this feature presents some risks which we discuss in Section 4. We recognize the thoughtful effort of Signal to provide a highly secure and private messaging solution, for example by not storing any user information [Sig21b, Sig21a], aiding deniability in practice.

1.1 Discussion on related work

In this work, we assume two parties execute a secure messaging protocol aided by a public key infrastructure and a (logical) server that routes messages between parties. Generally, messaging solutions build on two main components: an initial key exchange procedure and, upon initial authentication, the actual messaging. Signal implements these two phases using X3DH [MP16] and the Double Ratchet algorithm [PM16], respectively, where in the latter, keys are regularly updated, or ratcheted, to preserve security in the case of state exposure.

The conceptual simplicity of deniability hides multiple nuances, which lead to a plethora of definitions (cf. [DNS98, RGK06, BFG+22, CF11, DFG+13, HW21, HLLC11, VGIK20, RG05] for a non-exhaustive list[1]).

The usefulness of deniability has been debated before [CS20], for example in the context of OTR in 2014 [Hea14]. We discuss this matter in Section 4, and while some issues and arguments that we mention there have been raised before in the community, we hope to provide a more structured, up-to-date and thorough perspective compared to what has been done so far.

Two main flavours of deniability appear in the literature: *online* deniability [DKSW09] and *offline* deniability. Online deniability refers to the setting where the judge can interact with parties *during* the execution of the protocol, whereas in offline deniability the judge receives relevant

[1] Brendel et al. provide a relatively up-to-date survey on deniability in [BFG+22, Appendix A].

information only after all communication has ceased. Unger and Goldberg conjecture the incompatibility of online deniability with asynchronous key exchange protocols [UG15]. Moreover, it is questionably applicable in many practical scenarios. In court, evidence generally pertains to *past* events, so the scenario in which a judge actively colludes with a party to frame another seems unreasonable if the judge is not malicious and useless otherwise (since the judge can anyway rule against the victim). If the judge considers also current events and can mount a (remote) shoulder surfing attack, the human factor invalidates online deniability. The judge can instruct the incriminating party to perform an action that in practice frames the victim—for example, by asking a question that only the victim can answer. This suggests an analogy between online deniability and the Turing test [Tur50], but we leave it to future work to explore this parallelism. We therefore consider offline deniability in our model.

Vatandas et al. [VGIK20] analyze the deniability of the X3DH protocol. They prove that the protocol is offline deniable even assuming the incriminating party does not follow the prescribed protocol using knowledge of exponent-style assumptions [BP04] (which seem nevertheless necessary under their security notion). Thus, their simulator is "non-constructive" and its practicality is unclear.

All the formal and informal treatments of deniability we are aware of restrict the protocol transcript to be only the *cryptographic* transcript, i.e., the cryptographic material exchanged between parties. Albeit interesting from a cryptographic perspective, this approach limits the *practicality* of deniability since it ignores a plethora of additional information that can frame a party to a protocol execution. In the context of X3DH, partially signed prekey bundles seem to have been abstracted away in previous work on deniability [HKKP22, BFG+22], except for [VGIK20] that considers signature keys as *auxiliary input* given to the adversary in both the real and ideal cases in its definition of deniability.[2] This use of signatures precludes participation deniability and, in particular, strong forms of deniability considered, among others, in works that construct post-quantum X3DH variants [HKKP22, BFG+22].

Modern messaging solutions such as Signal build on a complex ecosystem of communication protocols, cryptographic solutions, and asynchronous multi-device approaches. The resulting complex stack gives a modern and usable messaging experience, but as we argue, this can (and, in the case of all solutions we are aware of, does) come at the cost of deniability.

In concurrent and independent recent work, Reitinger et al. [RMA+23] perform studies to investigate plausible deniability in a hypothetical legal trial setting. The authors develop a tool that allows participants to edit conversation transcripts: this is very similar to our proposal in this work, except that we propose such a feature could be integrated directly within the Signal application. When participants are first given the tool to interact with, they conclude that a party with transcript evidence incriminating them is guilty *less often* than those who are not given the tool, which seems to support a hypothesis that the awareness and ability to edit transcripts results in more tangible deniability.

2 Our model

In this section we introduce our deniability model, which analyzes deniability *in practice* by overcoming the limitations that we presented in the previous sections. Our model uses the real/ideal paradigm: the judge must differentiate between these worlds. Our model captures the interaction between two parties, Alice and Bob, and aims for offline deniability: Bob, the incriminating party, hands over his state to the judge *after* the protocol execution to frame the victim, Alice. We assume that both parties execute the protocol *honestly*. In particular, Bob does not devi-

[2] We note that auxiliary *input* is not sufficient to capture additional information, e.g., via the use of TLS on top of the core protocol, that is (adaptively) generated *during* protocol execution.

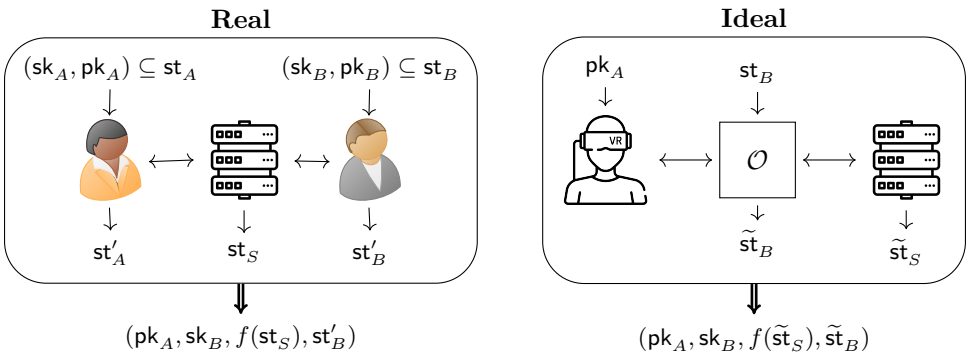

Figure 1: Our model for real world deniability in a two-parties protocol. The simulator is depicted by the virtual reality device. The application-specific oracle $\mathcal{O}$ models the ability of the simulator to modify Bob's state. Auxiliary data aux is also given to the judge in both the real and ideal worlds.

ate from the protocol to incriminate Alice. As we discussed in the previous section, achieving deniability in the malicious setting is very difficult, e.g., Bob could circumvent deniability with remote attestation [GPA19] or algorithm substitution attacks [AQ22]. We present our model in Fig. 1.

In the real world (on the left in Fig. 1), Alice and Bob execute the protocol by relying on an external server. The server is a logical entity: it can represent several physical machines in a centralized or distributed setting. Both parties input an initial state (st_A and st_B) that contains the corresponding long-term private and public keys. We do not force a type for a party's state: this can be anything, e.g., the entire phone. At the end of the execution the real world returns Alice's public key pk_A, Bob's secret key sk_B, a function of the server's internal state $f(\mathsf{st}_S)$, and Bob's final state st'_B. The function f models different leakages from the server. For example, a subpoena might force the maintainers to reveal some of the server's data, or the server can be completely breached, in which case f is the identity function.

The ideal world encompasses three actors: an oracle, a simulator and a server. The server is an exact replica of the server in the real world. The simulator inputs the victim's public key, i.e., Alice's pk_A, and interacts with the oracle to produce a simulation of Bob's final state. The goal of the simulator in collaboration with the oracle is to show that Bob can reach a simulated final state $\widetilde{\mathsf{st}}_B$ starting from an initial state st_B *without interaction with Alice* (i.e. without knowing st_A), where $\widetilde{\mathsf{st}}_B$ is indistinguishable (defined below) from the real final state st'_B. To this end, the oracle takes Bob's initial state st_B as input, interacts with the simulator and the server and, at the end of the experiment, outputs $\widetilde{\mathsf{st}}_B$. The simulator does not output the simulated state: state evolution is mediated through the oracle, which complies with the simulator's instructions by taking into account the practical constraints. The oracle models the simulator's practical effectiveness in simulating Bob's state and communicating with the server: the initial state might be encrypted with some unknown secret key and the server might require client's authentication. In a way, the oracle models the level of control Bob has on his client. For example, at one extreme, Bob may have rooted his phone and so is able to extract all secrets/messages contained within; in this case the oracle is very permissive. At the other extreme, Bob may know nothing about technology and so is limited to using the client to try to forge messages.

Fig. 2 shows two instantiations of the oracle $\mathcal{O}$: on the left without authentication and on the right with authentication. Both oracles input an operation $\mathsf{op} \in \{\mathsf{get}, \mathsf{set}, \mathsf{forward}, \mathsf{finish}\}$ and a corresponding $\mathsf{payload}$. The oracle that mimics authentication checks the payload's legitimacy

with verification function Vf before updating the state. The payload can be a message encrypted with TLS that is decrypted and stored in the state only if it comes from the correct server. The simulator interacts with the oracle to simulate Bob's state and communicate with the server while enforcing the authentication mechanism.

Oracle NO-AUTH(op, payload)	Oracle AUTH(op, payload)
1 : **if** op = get **then**	1 : **if** op = get **then return** $\perp$
2 : **send** st_B to simulator	2 : **elseif** op = set $\wedge$ $\mathsf{Vf}(\widetilde{\mathsf{st}}_B, \mathsf{payload})$ **then**
3 : **elseif** op = set **then**	3 : $\widetilde{\mathsf{st}}_B \leftarrow$ **update** $\widetilde{\mathsf{st}}_B$ according to payload
4 : $\widetilde{\mathsf{st}}_B \leftarrow$ payload	4 : **elseif** op = forward **then**
5 : **elseif** op = forward **then**	5 : **authenticate** to server using st_B
6 : **send** payload to server	6 : **send** payload to server
7 : **send** server's answer to simulator	7 : **send** server's answer to simulator
8 : **elseif** op = finish **then return** $\widetilde{\mathsf{st}}_B$	8 : **elseif** op = finish **then return** $\widetilde{\mathsf{st}}_B$
9 : **else return** $\perp$	9 : **else return** $\perp$

Figure 2: Two oracles for our deniability model. The NO-AUTH on the left models a system *without* authentication, the AUTH oracle on the right models *signature*-based authentication.

The ideal world outputs Alice's public key pk_A, Bob's secret key sk_B, some partial leakage of the server $f(\widetilde{\mathsf{st}}_S)$ for some function f, and Bob's simulated state $\widetilde{\mathsf{st}}_B$. The judge J, with the help of some auxiliary information—that is, any prior information the judge might have— must distinguish between the outputs of the two worlds. The judge rules *in dubio pro reo*: if the judge has doubts about Alice's guilt, she rules in favor of Alice.

Definition 1 *We say that a system using a server S coerced to reveal information that the function f represents is* deniable *if, for any victim A and any incriminating party B, there exists a simulator with access to an oracle $\mathcal{O}$, such that for any judge J with auxiliary information* aux *the following holds:*

$$\left|\Pr[J(\mathsf{aux}, \mathsf{pk}_A, \mathsf{sk}_B, f(\mathsf{st}_S), \mathsf{st}'_B)] - \Pr\left[J(\mathsf{aux}, \mathsf{pk}_A, \mathsf{sk}_B, f(\widetilde{\mathsf{st}}_S), \widetilde{\mathsf{st}}_B)\right]\right| \leq \nu,$$

where ν is a value that models the in dubio pro reo *approach: if the judge is* unsure*, for some amount of uncertainty, then she rules in favour of Alice. The exact value of ν depends on the context in which our definition is used.*

3 Signal: a case study

In this section we discuss the Signal application and its deniability. We focus on Signal because it offers the best security and privacy guarantees among deployed secure messaging solutions. Moreover, the cryptographic algorithms underlying Signal are deniable under some existing cryptographic notions of deniability [VGIK20, MP16, PM16].

We distinguish two cases: normal authentication and authentication with the sealed sender feature. The sources for this section are the Signal server source code [Sig22] and Signal blog post about sealed-sender-based authentication [Lun18].

3.1 Normal authentication

To send a message, the sender's client makes a POST request to the `/v1/messages/{receiver}` endpoint of the server with a payload that contains the encrypted message(s) and the receiver's

identifier. In addition, the server authenticates the sender using Dropwizard[3] basic authentication, which comprises the sender's identity and a password (a shared secret with the Signal server). Then, the server performs a few extra checks (e.g., the server enforces rate limiting) and creates an "envelope" containing the authenticated sender's identity, the encrypted message and a timestamp. The server forwards this envelope to the receiver or stores it until the receiver is online.

Deniability in practice. From the description above, we deduce that a message *delivered* by a legitimate and honest Signal client originates from the claimed sender, if the Signal server behaves as intended. This means that if someone inspects the receiver's device, they can be (very) confident that the message indeed comes from the sender. The only deniability argument left for the sender to make is to argue that the receiver tampered with the stored messages, i.e., the receiver edited the list of received messages in the local database. Depending on the kind of device, modifying the database is more or less challenging (e.g., it requires basic technical knowledge on the desktop client as the database encryption key is stored in clear[4], while advanced knowledge is required for mobile applications).

More formally, in our model, if there is authentication (AUTH oracle in Fig. 2), Signal is not deniable as Bob's state (which comprises the received set of messages) is updated only upon receiving a message from the Signal server certifying Alice sent it. This can happen only if Alice authenticated to the server to send a message, which the simulator cannot do as it does not know Alice's credentials. However, if the oracle NO-AUTH is used, the simulator can modify Bob's state to make it consistent with some messages received from Alice. Hence the deniability claim relies solely on the capability of the receiver to tamper with the local database.

3.2 Authentication with sealed sender

Authentication with the sealed sender feature conceals the identity of the sender from everyone except the receiver, *including* Signal servers. By default, this option is only enabled between two parties that are known to each other (e.g., contacts) and works as follows. Periodically, clients retrieve a certificate signed by Signal containing the client identity, phone number and an expiration date. In addition, each party derives some value called *delivery token* from their profile key and registers it with Signal. To send a message, the sender's client encrypts the certificate, the message and the sender's identity. The resulting ciphertext, the receiver's identity, and the delivery token are forwarded to the server using the same endpoint as in normal authentication. The server only checks that the token corresponds to the intended recipient, and forwards the ciphertext when the receiver is online; no authentication of the sender is performed. Finally, the receiver decrypts the ciphertext and checks that the certificate is valid and corresponds to the (claimed) sender's identity.

Deniability in practice. This setting offers brighter prospects for deniability than normal authentication. As the server never authenticates the sender, receiving a message from Alice that was correctly forwarded by the server does not mean she actually sent it. However, the fact that the message includes the sender's certificate hinders their capability to deny. Let's say the victim Alice claims Bob: (1) forged a message coming from her, then (2) made a Signal server forward that message to himself and finally (3) his client successfully delivered it. It means that Bob managed to recover one of Alice's certificates that is valid for that specific time frame. In turn, that implies that he (or an accomplice) recently received a message from Alice

[3] `https://github.com/dropwizard`

[4] Joshua Lund from Signal comments on this here `https://community.signalusers.org/t/vulnerabilities/4548/7`.

and extracted the certificate to forge a valid message. Therefore, in practice, one can be very confident that Alice exchanged messages with Bob recently, which weakens plausible deniability. Alice can still deny that she sent a particular message, but the claim relies solely on the fact that Bob had the technical knowledge to extract the certificate and forge a valid message or, as in normal authentication, the capacity of tampering with the messages stored on his client.

Transposing this argument in our model, we see that Signal with sealed sender is not deniable in the authenticated setting ($\mathcal{O} = \mathsf{AUTH}$) as Bob's state will accept Alice's message only if a legitimate certificate was provided. However, the simulator cannot obtain one from the server as it cannot authenticate as Alice. In the setting where Bob can freely modify the set of received messages ($\mathcal{O} = \mathsf{NO\text{-}AUTH}$), Signal with sealed sender is deniable.

3.3 Summary and discussion

In this section, we saw that the plausible deniability of Signal is practically non-existent unless the receiver has good technical capabilities. To summarise:

- Using normal authentication, if the client delivers a message from Alice, then it surely came from Alice because the server authenticated with her.
- Using sealed-sender-based authentication, if Bob's client delivers a message from Alice, then with good probability Bob received a message from Alice recently. Bob could forge a message coming from Alice by asking for Alice's certificate from ones of Alice's contacts, but we deem this to be relatively impractical.
- In both modes of authentication, deniability is plausible only if the receiver has good technical knowledge. If the incriminating party lacks technical knowledge, the judge will hardly believe the victim's denial of participation.

We conclude that, in most cases, the Signal application does not provide plausible deniability. It is interesting to see that the deniability claim becomes stronger as the system becomes easier to breach and vice versa. For instance, assuming a perfectly "secure" system (e.g., clients are impenetrable, participants can only interact through an authenticated client like in our model with $\mathcal{O} = \mathsf{AUTH}$) Signal is not deniable. Hence, we find ourselves in an undesirable position: some aspect of security clashes with deniability. We claim that the best way to solve this problem is to either give up on deniability or to propose "cracking" as a controlled feature. For example, Signal could include an edit/add button that allows the user to modify received messages or add new ones. This could lead to other problems, but we believe it is one of the few credible solutions to achieve practical plausible deniability. Overall, some form of plausible deniability can only be guaranteed if a practical, user-friendly simulator exists, which is currently not available. We explore this further in the next section.

4 Discussion

In this section we elaborate on our discussion of practical deniability and its consequences.

Deniability in front of the public. The leak of Hillary Clinton's email database [Wik16b] is often cited as a practical example in which deniability would have had an impact. Wikileaks published a set of emails (containing DKIM signatures on the content) and some of the authors claimed that they were tampered with. However, the signatures demonstrated, from the perspective of the general public, that the leaks were legitimate [Wik16a]: the emails were *not* deniable. Here the *general public* is the judge, instead of an official like a judge in a courtroom as usually envisioned. Our model suffices to capture this setting as it captures the real-life scenario where Bob's data are stolen (e.g., keys and transcripts), published on the internet and Alice wants to deny she ever wrote the published messages to Bob. Note that our model is actually stronger

than what is actually needed to model this scenario as the public (i.e. the judge) does not get Bob's secret key.

Deniability in court. The relevance of (message or cryptographic) deniability in court is debatable. In our research, we did not find any trial in which deniability was invoked (successfully or otherwise), even if electronic messages are often used as evidence. For instance, Signal messages (probably extracted from suspects' phones) were used in at least two recent high-profile cases in the US, seemingly without being contested (United States v. Rhodes et al.[5] and United States v. Jarret Crisler[6]). We also reviewed court cases mentioning WhatsApp messages in the jurisprudence of several Swiss Cantons (openly accessible, e.g. [dCdN22, dCdF22, dCdV22]). We had to settle for WhatsApp in our research as Signal is not mentioned in these databases. Even if WhatsApp is arguably much less deniable than Signal because of additional metadata that is stored and is visible to an adversary, we think that a court's opinion would be similar or the same for Signal or other messaging applications. This seems especially true if the court does not subpoena the relevant service provider for metadata, which is true for the Swiss cases we considered.

In the many occurrences of the keyword "whatsapp" that we found, the authenticity of messages was never questioned (except one case discussed below). We list some reasons here.

- Judges and lawyers are not aware of cryptographic deniability, let alone how the concept works. Therefore, if deniability is a desired feature, it needs to be simple and practical (e.g., with the possibility to edit the list of received messages).
- Communication transcripts are only a small part of the evidence and removing them might not make a difference. In addition, messages can usually be plausibly authenticated given the context and other data. For instance, in several civil or petty crime cases we reviewed, screenshots of conversations are considered without being contested. We only found one case [dCdF20] where a party in an appeal denied being the author of messages produced on a screenshot, but the matter was not taken further (for example by ordering analysis of the phones) as the judge deemed the messages to be credible and decided it would not change the original ruling. In more cryptographic terms, real and simulated transcripts/communication can be distinguished with good probability given auxiliary data.
- Judges feel the law is protected against forgeries as false testimony is a criminal offence. This implies that any transcript (even a very deniable one like a screenshot) is considered in court.

How to make a deniable system. Messaging services that focus on privacy like Signal use deniable cryptographic protocols (like X3DH [MP16]) and make it a selling argument, leading users to believe that deniability is a feature of the system. We showed in Section 3 that the situation is not so simple. We list here a few thoughts on the way deniability could be handled in the design of a messaging application.

- The first question is whether deniability is a necessary or desirable feature. As we saw, obtaining real deniability in practice implies non-trivial challenges that can impact users (e.g., spamming), performance (e.g., cost of adding deniability in the protocol, attacks on sealed-sender-based authentication [TLMR22]), the service (e.g., respectability) or security (e.g., authenticity is somewhat lost if Bob can easily forge messages from Alice).
- If deniability is a system goal, then one needs to think about the threat model: who are we defending against? For instance, who will be the judge, which capacity will the incriminating adversary (Bob) have, etc. If the threat model assumes a global adversary

[5] https://www.justice.gov/opa/press-release/file/1462481

[6] https://www.forbes.com/sites/thomasbrewster/2021/02/08/can-the-fbi-can-hack-into-private-signal-messages-on-a-locked-iphone-evidence-indicates-yes/

that *actively* colludes with Bob to frame a user, then maybe a messaging application that registers its users' identities (e.g., phone numbers) is not the most suitable tool and other services should be used instead (e.g., anonymous communication over Tor). The best way to avoid having to consider deniability in the first place is to guarantee anonymity.
- The system must provide *practical* deniability against the kind of adversaries that the threat model considers. In particular, deniability must be plausible for all, which means the "simulator" must be practical. For instance, as mentioned above, this can be achieved by offering a feature that allows users to edit their conversation history.
- The worst enemy of deniability is auxiliary data. The system should keep as little data as possible and remove it once it became useless, both on the servers and the clients.

Is practical deniability really what we want? One of the downsides of practical plausible deniability is that it could help the spread of "fake news" by making the fabrication of messages accessible to all. Also, in the jurisprudence we reviewed, messages voluntarily disclosed or retrieved by the police on victims' phones were used to convict criminals: practical deniability could invalidate these evidences. In addition, as pointed out by Jeff Burdges[7], there is also a risk that only rich and powerful people would have the resources to argue deniability in a court of law (or against the public opinion). Finally, practical deniability would also prevent a victim that receives abusive messages from proving the culpability of their author.

On the other hand, in some places there is a need for a way to communicate freely without fear of being held accountable. As an example, consider a political activist forced to hand their phone to the police. In this case, plausible deniability could help the other members of the group avoid charges on the account of the messages they sent.

This discussion is reminiscent of the debate on end-to-end encryption, but the impact of deniability is less clear. Encryption solves, among other things, the ubiquitous problem of mass surveillance: an attacker has no access to messages in transit. Encryption is a mathematically-based property that has an immediate and global effect. By contrast, deniability is a more subtle, subjective property, which involves important human and societal factors that can vary from case to case: which auxiliary information the judge has, or which distinguishing probability can protect—or frame—the victim. Since deniability is not used in practice, it is difficult to argue about its impact, but we find that a community-wide discussion in this sense is necessary.

Also, as hinted above, one can wonder if the property that we are looking for is not simply anonymity. By making it hard for the judge to track down (and thus identify, profile and obtain data about) the sender, or to link some account to them, we might avoid the deniability question altogether in several settings. All in all, aiming for plausible deniability in general public messaging applications like WhatsApp and Signal, which link accounts to identities (e.g., phone numbers) seems, if not hopeless, very difficult to achieve. Even if adding the possibility to edit/add messages to a conversation would be a first step towards that goal, it is unclear whether this would be accepted as deniable by the society.

Finally, from a cryptographic perspective, deniable protocols involve complex primitives like ring signatures and zero-knowledge proofs [UG18]. This is also true in the post-quantum setting, where the post-quantum variants of X3DH [BFG+22, HKKP22] use designated verifier or ring signature schemes. Therefore, cryptographic deniable protocols have consequences on both efficiency and security (particularly when based on non-standard primitives that are easier to misimplement). Hence, these protocols should only be integrated in practically deniable systems to have any *positive* effect.

Acknowledgments. We thank Sylvain Chatel, Vero Estrada-Galiñanes, Abdullah Talayhan and the RWC 2023 reviewers for their comments on this document and colleagues at EPFL for

[7] `https://mailarchive.ietf.org/arch/msg/mls/Kk6qai3kEza8B-L-_5R-qsEjtK0/`

their feedback on the corresponding presentation.

References

[AQ22] Marcel Armour and Elizabeth Quaglia. Subverting Deniability. In *ProvSec*, 2022.

[BFG+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake. In *PKC*, 2022.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES*, 2004.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, 2004.

[CF11] Cas Cremers and Michèle Feltz. One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability. *IACR Cryptol. ePrint Arch.*, 2011.

[CHMR23] Suvradip Chakraborty, Dennis Hofheinz, Ueli Maurer, and Guilherme Rito. Deniable authentication when signing keys leak. Cryptology ePrint Archive, Paper 2023/213, 2023. `https://eprint.iacr.org/2023/213`.

[CS20] Sofía Celi and Iraklis Symeonidis. The current state of denial, 2020.

[dCdF20] Tribunal Cantonal du Canton de Fribourg. Arrêt de la Chambre Pénale du Tribunal Cantonal N° 502 2020 76. `https://publicationtc.fr.ch/?dec=2fedf4e8c4b54ca8b5c1d5efcfa2972b&index=TC`, 2020. Last visited on 07-11-2022 (in French).

[dCdF22] Tribunal Cantonal du Canton de Fribourg. Jurisprudence du Tribunal Cantonal. `https://publicationtc.fr.ch/`, 2022. Last visited on 07-11-2022 (in French).

[dCdN22] Tribunal Cantonal du Canton de Neuchâtel. Jurisprudence du Tribunal Cantonal. `https://jurisprudence.ne.ch`, 2022. Last visited on 07-11-2022 (in French).

[dCdV22] Tribunal Cantonal du Canton de Vaud. Jurisprudence du Tribunal Cantonal. `https://www.findinfo-tc.vd.ch/justice/findinfo-pub/internet/`, 2022. Last visited on 07-11-2022 (in French).

[DFG+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A Cryptographic Analysis of OPACITY - (Extended Abstract). In *ESORICS*, 2013.

[DKSW09] Yevgeniy Dodis, Jonathan Katz, Adam D. Smith, and Shabsi Walfish. Composability and On-Line Deniability of Authentication. In *TCC*, 2009.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent Zero-Knowledge. In *STOC*, 1998.

[GPA19] Lachlan J. Gunn, Ricardo Vieitez Parra, and N. Asokan. Circumventing cryptographic deniability with remote attestation. *PoPETs*, 2019.

[Gre14] Matthew Green. Noodling about im protocols. `https://blog.cryptographyengineering.com/2014/07/26/noodling-about-im-protocols/`, 2014. Last visited on 02-11-2022.

[Hea14] Mike Hearn. Value of deniability (mailing list discussion). `https://moderncrypto.org/mail-archive/messaging/2014/001173.html`, 2014. Last visited on 09-11-2022.

[HKKP22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for signal's handshake (x3dh): post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 35(3):1–78, 2022.

[HLLC11] Lein Harn, Chia-Yin Lee, Changlu Lin, and Chin-Chen Chang. Fully deniable message authentication protocols preserving confidentiality. *Comput. J.*, 54(10):1688–1699, 2011.

[HW21] Andreas Hülsing and Florian Weber. Epochal signatures for deniable group chats. In *S&P*, 2021.

[Lun18] Joshua Lund. Technology preview: Sealed sender for signal. `https://signal.org/blog/sealed-sender/`, 2018. Last visited on 03-11-2022.

[Mar13] Moxie Marlinspike. Simplifying OTR deniability. `https://signal.org/blog/simplifying-otr-deniability/`, 2013. Last visited on 25-10-2022.

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. `https://www.signal.org/docs/specifications/x3dh/`, 2016. Last visited on 25-10-2022.

[PM16] Trevor Perrin and Moxie Marlinspike. The Double Ratchet Algorithm. `https://signal.org/docs/specifications/doubleratchet/`, 2016. Last visited on 25-10-2022.

[RG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In *CCS*, 2005.

[RGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In *CCS*, 2006.

[RMA+23] N. Reitinger, N. Malkin, O. Akgul, M. L. Mazurek, and I. Miers. Is cryptographic deniability sufficient? non-expert perceptions of deniability in secure messaging. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1658–1676, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.

[Sig21a] Signal. Grand jury subpoena for Signal user data, Central District of California (again!). `https://signal.org/bigbrother/cd-california-grand-jury/`, 2021. Last visited on 25-10-2022.

[Sig21b] Signal. Grand jury subpoena for Signal user data, Central District of California. `https://signal.org/bigbrother/central-california-grand-jury/`, 2021. Last visited on 25-10-2022.

[Sig22] Signal. Signal-server. `https://github.com/signalapp/Signal-Server`, 2022.

[TLMR22] Nirvan Tyagi, Julia Len, Ian Miers, and Thomas Ristenpart. Orca: Blocklisting in {Sender-Anonymous} messaging. In *USENIX Security*, 2022.

[Tur50] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.

[UG15] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In *CCS*, 2015.

[UG18] Nik Unger and Ian Goldberg. Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging. In *PoPETs*, 2018.

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the Cryptographic Deniability of the Signal Protocol. In *ACNS*, 2020.

[Wik16a] WikiLeaks. DKIM Verification. `https://wikileaks.org/DKIM-Verification.html`, 2016. Last visited on 07-11-2022.

[Wik16b] WikiLeaks. Hillary Clinton Email Archive. `https://wikileaks.org//clinton-emails/`, 2016. Last visited on 07-11-2022.