How to kickstart $\mathcal{F}_{\mathrm{SMT}}^{S\to R}$ with Short Authentication Strings and Out-Of-Band Communication

Wasilij Beskorovajnov[3] and Jörn Müller-Quade[1,2,3]

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2] KASTEL Security Research Labs, Karlsruhe, Germany
mueller-quade@kit.edu
[3] FZI Research Center for Information Technology, Karlsruhe, Germany
Beskorovajnov@fzi.de

Abstract. We study "send this data to that device now" exchanges under an active network adversary without PKI or pre-shared secrets. We model a human-verifiable out-of-band channel for comparing short codes as a first-class UC functionality $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$. Building on this, we give two commitment-style protocols, π_{SAS} (one-sided) and $\pi_{\mathrm{SAS}}^{\times}$ (mutual), that are new variations of MANA IV and prove that they realize $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ and $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ with explicit misbinding parameter $\varepsilon = 2^{-t}$. We then compose SAS-based authentication with standard KEM/DEM encryption to obtain a UC-secure message-transfer functionality $\mathcal{F}_{\mathrm{SMT}}^{S\to R}(\varepsilon)$, preserving the explicit ε under composition, and we detail practical instantiations over signatures or MACs. Complementing the theory, we systematize real-world tooling: popular file-transfer utilities either form unauthenticated WebRTC/DTLS channels or use PAKE "one-code" designs that couple rendezvous and a long shared secre, but none deploy a session-bound SAS. Our approach decouples rendezvous from authentication and reduces the out-of-band burden to comparing a short t-bit string. We also sketch an RO-free variant (coin-flip plus non-malleable commitments) with the same user interface.

1 Introduction

There is a small but active ecosystem of "send this to that device now" tools: PAKE/one-code utilities such as Magic Wormhole[4], WebWormhole[5], and croc[6]; browser-based WebRTC/DTLS tools like PairDrop[7] and FilePizza[8]. Despite different plumbing, all must answer the same question: is this the intended peer under an active network adversary?

[4] https://github.com/magic-wormhole/magic-wormhole
[5] https://github.com/saljam/webwormhole
[6] https://github.com/schollz/croc
[7] https://github.com/schlagmichdoch/PairDrop
[8] https://github.com/kern/filepizza

A first observation is that today's file-transfer tools do not deploy a session-bound, human comparison step. Instead we see two dominant patterns. (i) Unauthenticated DTLS over WebRTC: the browser checks a DTLS certificate against a fingerprint delivered by signaling, but there is no user-verifiable bind, so a malicious signaling service can splice two DTLS associations and sit in the middle; (ii) PAKE "one-code" designs: a single human-readable code carries both a rendezvous reference and a PAKE password. This yields smooth UX but requires moving a comparatively long codes out of band and shifts security to the secrecy and integrity of that code.

We revisit a different, well-studied bind: the *short authentication string* (SAS). Here each endpoint derives a short t-bit value from the live run (e.g., via a commitment-style transcript hash or a coin-flip bound to the transcript) and the user compares the two values over an auxiliary authenticated channel. Vaudenay showed that when users compare correctly, an active man-in-the-middle succeeds with probability at most 2^{-t} [34]. This paper treats that human-verifiable step as a first-class component, models it in UC, and shows how SAS-based authentication composes to secure message transfer while keeping the out-of-band burden to comparing only a short t-bit string.

Threat Model and Scope Following the standard SAS and device-pairing literature, we work in a two-party setting with honest endpoints S and R, while the adversary has full control over the network and the rendezvous/service infrastructure. We therefore focus on the case of a channel/service adversary with honest endpoints, and make this explicit in our UC functionalities. Modeling endpoint compromise (e.g., malicious devices or UIs) is orthogonal to our goals here and is discussed as a limitation and direction for future work.

Contributions

1. **Ideal OOB/SAS functionality.** We introduce $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$, a UC ideal functionality for a hybrid, human-verifiable out-of-band channel that authenticates short strings. The current version handles synchronous comparisons.
2. **PISAS and PISAS$^{\times}$ as UC secure variants of MANA IV.** We give protocols π_{SAS} (one-sided) and π_{SAS}^{X} (mutual) that follow the MANA IV commit–challenge–open mechanics [28, 25] (both parties confirm the SAS) while binding arbitrary messages (e.g., KEM public keys/encapsulations or context) with two differences.
   - First, we add session identifiers to the protocol description, which are crucial for UC security.
   - Second, we omit the initial plaintext message from the first network flow. This change is not required for UC, nor does sending the message inside the first flow compromise UC security. Our justification is that (A) the message is already included in the commitment, and the final flow (which also authenticates all flows) contains the opening information needed to recover it, and (B) we find it more sensible to reveal the actual message at the end, where it is immediately followed by authentication. In fact, some

commitment implementations might be more efficient if the message is not included.

We prove they UC-realize $\mathcal{F}_{\text{AUTH}}^{S\to R}(\varepsilon)$ and $\mathcal{F}_{\text{AUTH}}^{S\leftrightarrow R}(\varepsilon)$, respectively. We also present a sketch of a variant of π_{SAS}^{X} without random oracles in the full version.

3. **Composable secure message transfer.** We show how composing SAS-based authentication with standard encryption primitives (KEM/DEM) yields a UC realization of secure message transfer $\mathcal{F}_{\text{SMT}}^{S\to R}(\varepsilon)$, preserving the explicit misbinding parameter under composition. We detail two practical instantiations: an SMT variant over signatures and an SMT variant over MACs in the Appendix B.
4. **Systematization of practice.** We systematize real-world tooling along the axes "how peers meet," "what the human binds," and "what the service learns" under a malicious-server assumption.
   In Section 6.1 we analyze deployed one-shot file transfer tools and show that many DTLS-/WebRTC-based designs provide no end-to-end authentication at all, and can be straightforwardly upgraded to our UC-secure pattern by adding a single SAS check.
   In Section 6.2 we survey deployed SAS usage and make explicit a separation between *session-bound SAS* (e.g., ZRTP, Matrix SAS, BLE Numeric Comparison) and *identity-key fingerprints* (e.g., Signal, WhatsApp, iMessage, Wire, Threema, Telegram), and use this to classify which deployed protocols are already subsumed by our analysis and which would require additional mechanisms.

The code and the reference application of the protocols π_{SAS}^{X} are available on Github[9].

2 Related Work

A well-known and widely adopted paradigm for establishing a session key, that can be used for secure file transfer, is an Authenticated Key-Agreement (AKA). These protocols, however, tend to be complex, both in terms of their design and their target guarantees. They are typically built for environments with multiple parties that may act interchangeably as sender or receiver, and with corruption that can be static or adaptive. This complexity is not itself a showstopper. The core challenge, however, lies in the authentication component of most AKA protocols, which usually relies on a public-key infrastructure (PKI), an assumption that is not viable in our setting.

It is worth noting that there is a line of AKA protocols where authentication is achieved via Short Authentication Strings (SAS), which can be seen—roughly speaking—as fingerprints of the communication transcript. As shown by Vaudenay [34], such mechanisms can securely realize the authentication step of an AKA

[9] `https://github.com/collapsinghierarchy/noisytransfer-protocol`

under a weak, human-verifiable out-of-band (OOB) channel. We will return to SAS-based approaches below.

Another nearby alternative is Password-Authenticated Key Exchange (PAKE), which requires a password distribution phase. This phase is often avoided in discussions, but it implicitly assumes the existence of some secure channel, oftentimes an OOB channel with confidentiality and authenticity, for the safe transmission or registration of the password. This is precisely the kind of assumption we wish to avoid. Instead, we assume only an authenticated OOB channel for "short strings", which removes the need for confidentiality guarantees that PAKE's registration phase relies on.

A closely related device-onboarding scenario appears with low-power/IoT devices, where unassociated devices share no state and PKI setup may be costly or infeasible. To address this, works in the *Message Recognition Protocol* (MRP) family introduce a weaker primitive that authenticates message origin without providing full identity authentication or confidentiality. MRPs can bootstrap secure links in such constrained settings [29, 19, 21, 11]. These are adjacent to our goal but target a different abstraction and guarantee set.

SAS and out-of-band (OOB) channels. Authenticating a session by comparing a short authenticated string (SAS) over an auxiliary, human-verifiable channel originates with Vaudenay [34], who formalized SAS-based authentication and gave commitment-style protocols that remain secure against attackers who may delay, drop, or replay the OOB message but can only cause a misbinding with probability at most 2^{-t} for t-bit SAS. Subsequent work optimized and generalized SAS for pairing and key agreement: Laur–Nyberg's MANA/MA-DH and MANA-IV families [25], Pasini–Vaudenay's non-interactive message authentication [31], and group extensions of SAS-based AKE [27]. In particular, Laur and Pasini's consolidated treatment [28] describes the MANA IV pattern, i.e., commitment, challenge, and opening, followed by a bidirectional SAS comparison, for mutual data authentication.

Formal security for SAS protocols. Most of the SAS literature proves security in bespoke, game-based models tailored to pairing/AKE (typically commitment-based, sometimes in the random-oracle model), including SAS-based message authentication and AKE with key-reuse constraints [26, 32, 31, 22]. These works bound an active attacker's success by 2^{-t} for a t-bit display and provide concrete, protocol-specific guarantees. However, to the best of our knowledge there is no treatment that (i) defines a UC authenticated-channel ideal functionality with an explicit misbinding parameter ε, (ii) shows that an SAS-based protocol realizes such a functionality in a modern UC framework, and (iii) composes this into higher-level secure message transfer while preserving the explicit ε under composition. A related line (e.g., [28]) studies user-aided security outside the modern UC framework and does not expose the OOB/SAS channel as a first-class ideal functionality with parameter ε. Closer to modern UC, Benin–Toledo–Tromer [11] model message-recognition protocols in UC using an ideal comparison primitive and focus on authentication bootstrapping (one human check, then continue)

without confidentiality (termed "privacy" there). They also do not parameterize the residual MitM guess probability as an explicit ε carried through composition. This is the gap our $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$, $\mathcal{F}_{\text{AUTH}}^{S\to R}/\mathcal{F}_{\text{AUTH}}^{S\leftrightarrow R}$, and $\mathcal{F}_{\text{SMT}}^{S\to R}$ treatment fills.

3 Preliminaries

3.1 Notation

We model $H : \{0,1\}^* \to \{0,1\}^n$ as a random oracle; for $t \le n$ let $\mathsf{Trunc}_t : \{0,1\}^n \to \{0,1\}^t$ be fixed-bit truncation. We write $\text{sas}_t(x) = \mathsf{Trunc}_t(H(x))$. $\mathsf{Ber}(\varepsilon)$ denotes a Bernoulli random variable with success probability ε. For a finite set S, we write $x \leftarrow\$ S$ to denote that x is sampled uniformly at random from S.

3.2 Cryptographic Primitives

- $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ is an **IND-CPA–secure** key-encapsulation mechanism.
- $\mathsf{DEM} = (\mathsf{Enc}, \mathsf{Dec})$ is an **OT-CPA–secure** data-encapsulation (symmetric) cipher.
- $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is an **EUF-CMA–secure** digital-signature scheme.

One-time IND-CPA security for a DEM. Let $\mathsf{DEM} = (\mathsf{ENC}, \mathsf{DEC})$ be a symmetric encryption scheme with keyspace $\mathcal{K}$. We define the one-time indistinguishability experiment against an adversary $\mathcal{A}$:

Definition 1 (OT-CPA for randomized DEM [23]). *In experiment* $\mathsf{Exp}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}}(\mathcal{A})$*:*

1. *The challenger samples* $K \leftarrow\$ \mathcal{K}$ *and* $b \leftarrow\$ \{0,1\}$.
2. $\mathcal{A}$ *outputs two equal-length messages* (m_0, m_1)
3. *The challenger samples fresh internal randomness* r *for* ENC *and returns* $c \leftarrow \mathsf{ENC}_K(m_b; r)$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs a bit* b'.

The advantage is

$$\mathsf{Adv}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}}(\kappa) \;:=\; \max_{\mathcal{A}} \left| \Pr[\mathsf{Exp}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}}(\mathcal{A})=1] - \tfrac{1}{2} \right|,$$

where $\mathsf{Exp}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}}(\mathcal{A})=1$ *iff* $b'=b$. *Security requires* $\mathsf{Adv}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}}(\kappa) = \mathsf{negl}(\kappa)$.

Commitments

Definition 2 (Commitment scheme). *A commitment scheme is a tuple of PPT algorithms* $(\mathsf{Commit}, \mathsf{Verify})$ *with message space* $\mathcal{M}$ *and randomness space* $\mathcal{R}$*, where*

$$(C, d) \;\leftarrow\; \mathsf{Commit}(m; r)$$

on input $m \in \mathcal{M}$ and randomness $r \leftarrow\$ \mathcal{R}$ produces a commitment C and decommitment witness d. The deterministic $\mathsf{Verify}(C, m, d)$ outputs a bit in $\{0,1\}$, accepting valid openings.

Correctness. *For all $m \in \mathcal{M}$,*

$$\Pr_{r \leftarrow\$ \mathcal{R}}\Big[\mathsf{Verify}(C, m, d) = 1 \;\Big|\; (C, d) \leftarrow \mathsf{Commit}(m; r)\Big] \;\geq\; 1 - \mathsf{negl}(\lambda),$$

i.e., honestly generated commitments open successfully except with negligible probability.

Computational binding. *For every PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{matrix}(C, m, m', d, d') \leftarrow \mathcal{A}(1^\lambda) : \; m \neq m' \\ \mathsf{Verify}(C, m, d) = 1 \wedge \mathsf{Verify}(C, m', d') = 1\end{matrix}\right] \;\leq\; \mathsf{negl}(\lambda).$$

That is, no efficient adversary can produce a single commitment along with two different valid openings.

Computational hiding. *For every PPT adversary $\mathcal{A}$, the advantage in the following experiment is negligible:*

1. *$\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$ of equal length.*
2. *A random bit $b \leftarrow\$ \{0,1\}$ and randomness $r \leftarrow\$ \mathcal{R}$ are chosen, and $C \leftarrow \mathsf{Commit}(m_b; r)$ is computed.*
3. *$\mathcal{A}$ is given C and outputs a guess b'.*

The hiding advantage is

$$\left|\Pr[b' = b] - \tfrac{1}{2}\right| \;\leq\; \mathsf{negl}(\lambda).$$

Thus, commitments to m_0 and m_1 are computationally indistinguishable.

In our main constructions we instantiate Commit as hash-then-open in the random-oracle model (binding and computationally hiding under [10]).

3.3 Universal Composability Framework

The UC framework [14, 13] additionally introduces an interactive Turing machine $\mathcal{Z}$, called the environment, which has to distinguish the real world execution from the ideal world simulation by actively communicating with the adversary during the function computation. Additionally, we define the environment's view with an adversary and a function as the input, the exchanged messages between the corrupted parties and adversary together with the outputs. This restricts the simulation-based proof since rewinding of the adversary is not possible in this case. For a formal statement, we recite the simulation paradigm of the UC framework by Canetti.

Definition 3 ([14]). *Let π be a protocol on inputs $\mathbf{x} = (x_1, \ldots, x_n) \in X^n$ for n parties and $\mathcal{F}$ a functionality. We say that π UC-securely realizes $\mathcal{F}$ if for all polynomially bounded adversaries $\mathcal{A}$ a simulator $\mathcal{S}$ exists such that for every polynomially bounded environment $\mathcal{Z}$:*

$$IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\lambda,\mathbf{x})_{\lambda\in\mathbb{N},\mathbf{x}\in X^n} \overset{c}{\approx} REAL_{\pi,\mathcal{A},\mathcal{Z}}(\lambda,\mathbf{x})_{\lambda\in\mathbb{N},\mathbf{x}\in X^n},$$

where $REAL_{\pi,\mathcal{A},\mathcal{Z}}(\lambda,\mathbf{x})_{\lambda\in\mathbb{N},\mathbf{x}\in X^n}$ *is the environment's view when interacting with* $P_1,\ldots,P_n$ *and* $\mathcal{A}$ *and* $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\lambda,\mathbf{x})_{\lambda\in\mathbb{N},\mathbf{x}\in X^n}$ *is* $\mathcal{Z}$*'s view when interacting with* $\mathcal{F}$ *and* $\mathcal{S}$*.*

Additionally, the Universal Composition framework has the property that any UC-secure protocol can be securely (concurrently) composed with any other UC-secure protocol. This is the universal composition theorem, which is rephrased here without proof.

Theorem 1 (UC composition theorem, see [14]). *Let* $\mathcal{F}$ *and* $\mathcal{G}$ *be two ideal functionalities. Further let* ρ *be a protocol that securely UC-realizes* $\mathcal{G}$ *and let* π *be a protocol that securely UC-realizes* $\mathcal{F}$*. Then the composed protocol* ρ^π *securely UC-realizes* $\mathcal{G}$*.*

We refer the proof to [14].

3.4 Ideal Functionalities

The following authenticated channel is included only as a warm-up: it motivates the first step in proving a simplified variant of our SAS-based authentication protocol. It is the standard $\mathcal{F}_{\text{AUTH}}^{S\to R}$ from Canetti [14], extended with a bounded active-forgery hook to capture the residual success probability from random guessing a short authentication string. Concretely, in $\mathcal{F}_{\text{AUTH}}^{S\to R}(\varepsilon)$ we let the adversary (i) schedule delivery and (ii) make a single online forgery attempt per session that succeeds with probability ε (e.g., $\varepsilon = 2^{-t}$ for a t-bit SAS). This abstraction serves as a stepping stone toward our cross-authentication functionality and the full SAS-based secure channel.

Compromise of Endpoints We ignore party corruptions in our two-party setting. Some works, such as [32], explicitly incorporate corruption of one endpoint in their models of authentication protocols and prove security as long as at least one endpoint remains honest and the corrupted endpoint continues to behave honestly. In our main protocols π_{SAS} and π_{SAS}^X, which realize the authentication functionalities $\mathcal{F}_{\text{AUTH}}^{S\to R}$ and $\mathcal{F}_{\text{AUTH}}^{S\leftrightarrow R}$, an adversary who corrupts a party gains at most its internal state. Since we do not store any long-term secrets in the state, our results are compatible with those works even without an explicit corruption model. The situation is different for protocols realizing $\mathcal{F}_{\text{SMT}}^{S\to R}$, where learning the internal state also reveals the secret keys corresponding to public keys previously authenticated by π_{SAS}^X, which trivially breaks confidentiality. We argue, however, that this honest-endpoint threat model is still in accordance with SAS-related work such as [32], which use the classical Bellare–Rogaway model [9], where security is defined for sessions between honest endpoints and full endpoint compromise is not within the scope of the guarantees. Extending our modeling to capture post-compromise security is an interesting direction for future work.[10]

[10] See, for example, recent analyses of post-compromise security for secure messaging protocols [17]

Ideal Functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$

Parameters. Forgery bound $\varepsilon \in [0,1]$.
Parties. A sender S, a receiver R, and the adversary $\mathcal{A}$. We assume S and R remain honest.
State per sid: $m \leftarrow \bot$; $R \leftarrow \bot$; $\mathit{forged}_{\mathsf{sid}} \in \{\bot, \mathsf{false}, \mathsf{true}\}$ (init $\bot$).

- **Delayed Send.** Upon input $(\mathsf{Send}, \mathsf{sid}, R, m)$ from S, record (m, R), set $\mathit{forged}_{\mathsf{sid}} \leftarrow \bot$, and send back-door $(\mathsf{Sent}, \mathsf{sid}, S, R, m)$ to $\mathcal{A}$.
- **Active Forgery (one online attempt).** Upon back-door $(\mathsf{Forge}, \mathsf{sid}, m', R)$ from $\mathcal{A}$ (at most once per sid): sample $b \leftarrow \mathsf{Ber}(\varepsilon)$; send $(\mathsf{ForgeResult}, \mathsf{sid}, b)$ to $\mathcal{A}$; set $\mathit{forged}_{\mathsf{sid}} \leftarrow (\mathsf{true}$ if $b{=}1$ else $\mathsf{false})$; if $b{=}1$, also record (m', R) for this sid.
- **Release.** Upon back-door $(\mathsf{ok}, \mathsf{sid})$ from $\mathcal{A}$:
  - If $\mathit{forged}_{\mathsf{sid}} = \mathsf{true}$, output $(\mathsf{Sent}, \mathsf{sid}, S, m')$ to R.
  - Else if $\mathit{forged}_{\mathsf{sid}} = \bot$ (no attempt), output $(\mathsf{Sent}, \mathsf{sid}, S, m)$ to R.
  - Else ($\mathit{forged}_{\mathsf{sid}} = \mathsf{false}$; attempted and failed), **output nothing**.
- **Ignore** all other inputs and back-door messages.

Cross-Authentication with a Bernoulli Forgery-Gate In the next step we lift $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ to a two-sided, user-aided cross-authentication functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ in the sense of Laur–Pasini [28]: each party contributes a short context string (e.g., ephemeral identifiers or public parameters), the adversary receives the prescribed leakage and may schedule delivery, and is granted a single online misbinding attempt that succeeds with probability ε (capturing random SAS guessing). On acceptance, both parties obtain the peer's authenticated contribution, which we later use as input to the SAS-based secure channel.

Ideal Functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$

Parameters. Forgery bound $\varepsilon \in [0,1]$.
Parties. Honest S and R, and the adversary $\mathcal{A}$.
State per sid. $m_S \leftarrow \bot$, $m_R \leftarrow \bot$; $\mathsf{forged} \in \{\bot, \mathsf{true}, \mathsf{false}\}$ (init $\bot$); $m_S^{\mathrm{fg}}, m_R^{\mathrm{fg}} \leftarrow \bot$; $\mathsf{delivered} \leftarrow \mathsf{false}$.

- **Propose (from S).** On $(\mathsf{Propose}, \mathsf{sid}, R, m_S)$: record m_S; send back-door $(\mathsf{Seen}, \mathsf{sid}, S, m_S)$ to $\mathcal{A}$.
- **Respond (from R).** On $(\mathsf{Respond}, \mathsf{sid}, S, m_R)$: record m_R; send back-door $(\mathsf{Seen}, \mathsf{sid}, R, m_R)$ to $\mathcal{A}$.
- **Active Forgery (one online attempt).** At any time *before* delivery, upon back-door $(\mathsf{Forge}, \mathsf{sid}, m'_S, m'_R)$ (at most once per sid): sample $b \leftarrow \mathsf{Ber}(\varepsilon)$; send $(\mathsf{ForgeResult}, \mathsf{sid}, b)$ to $\mathcal{A}$. If $b = 1$, set $(m_S^{\mathrm{fg}}, m_R^{\mathrm{fg}}) \leftarrow (m'_S, m'_R)$ and $\mathsf{forged} \leftarrow \mathsf{true}$; else set $\mathsf{forged} \leftarrow \mathsf{false}$.
- **Release.** Upon back-door $(\mathsf{ok}, \mathsf{sid})$ from $\mathcal{A}$, if $\mathsf{delivered} = \mathsf{false}$ and $m_S \neq \bot$ and $m_R \neq \bot$ then:

- If $\mathsf{forged} = \mathsf{true}$, output $(\mathsf{XAuth}, \mathsf{sid}, S, m_S^{\mathrm{fg}})$ to R and $(\mathsf{XAuth}, \mathsf{sid}, R, m_R^{\mathrm{fg}})$ to S; set $\mathsf{delivered} \leftarrow \mathsf{true}$.
- Else if $\mathsf{forged} = \perp$ (no attempt), output $(\mathsf{XAuth}, \mathsf{sid}, S, m_S)$ to R and $(\mathsf{XAuth}, \mathsf{sid}, R, m_R)$ to S; set $\mathsf{delivered} \leftarrow \mathsf{true}$.
- Else ($\mathsf{forged} = \mathsf{false}$; attempted and failed), **output nothing** and set $\mathsf{delivered} \leftarrow \mathsf{true}$.

• **Ignore** all other inputs/back-door messages.

Secure Message Transfer with a Bernoulli MitM-Gate The next functionality refines secure message transmission to account for the non-negligible online success of an active adversary under SAS/PAKE-style human authentication, in the same spirit as our authenticated-channel variants. We constrain leakage to message length only, i.e., $\ell(m) = |m|$. Classical UC $\mathcal{F}_{\mathrm{SMT}}^{S\to R}$ delivers the sender's message intact once the adversary schedules delivery, which overidealizes SAS/PAKE-backed channels: an attacker may achieve a *MitM misbinding* with probability essentially $\varepsilon \approx 2^{-t}$ for a t-bit SAS (up to standard concurrency factors), or roughly $1/|\mathcal{D}|$ per online guess in the PAKE setting [34, 32, 18]. Following Canetti–Halevi–Katz's UC treatment of low-entropy secrets [18], we adopt the "explicit guessing power" modeling and build a single Bernoulli(ε) gate into the functionality. Concretely, the adversary gets exactly one online misbinding attempt per session, reflecting a single SAS comparison; if it succeeds, confidentiality and authenticity fail for that session (the original m is learned) and the adversary may substitute the delivered message, with delivery still gated by $(\mathsf{ok}, \mathsf{sid})$.

Functionality $\mathcal{F}_{\mathrm{SMT}}^{S\to R}(\varepsilon)$

Parameters. Misbinding success bound $\varepsilon \in [0, 1]$; leakage $\ell(m) = |m|$.
Parties. Honest S and R, and the adversary $\mathcal{A}$.
State per sid: $m \leftarrow \perp$; $R \leftarrow \perp$; $\mathsf{auth} \in \{\perp, \mathsf{true}, \mathsf{false}\}$ (init $\perp$); $m^{\mathrm{mitm}} \leftarrow \perp$; $\mathsf{delivered} \leftarrow \mathsf{false}$.

• **Send.** On $(\mathsf{Send}, \mathsf{sid}, R, m)$ from S: record m and R; set $\mathsf{auth} \leftarrow \perp$ and $m^{\mathrm{mitm}} \leftarrow \perp$; send back-door $(\mathsf{Sent}, \mathsf{sid}, S, R, \ell(m))$ to $\mathcal{A}$.

• **Active MitM.** Before delivery and at most once per sid, upon back-door $(\mathsf{MisbindTry}, \mathsf{sid})$: sample $b \leftarrow \mathsf{Ber}(\varepsilon)$; send $(\mathsf{MisbindResult}, \mathsf{sid}, b)$ to $\mathcal{A}$.
- If $b = 1$, set $\mathsf{auth} \leftarrow \mathsf{true}$ and send back-door $(\mathsf{Reveal}, \mathsf{sid}, m)$ to $\mathcal{A}$.
- Else set $\mathsf{auth} \leftarrow \mathsf{false}$.

• **Set substituted payload.** At any time before delivery, upon back-door $(\mathsf{SetMitmPayload}, \mathsf{sid}, m')$: if $\mathsf{auth} \neq \mathsf{true}$ then ignore; else require $|m'| = |m|$ (else ignore); if $m^{\mathrm{mitm}} = \perp$ then set $m^{\mathrm{mitm}} \leftarrow m'$.

• **Delivery (gated by ok).** Upon back-door $(\mathsf{ok}, \mathsf{sid})$ from $\mathcal{A}$, if $\mathsf{delivered} = \mathsf{false}$ then:
- If $\mathsf{auth} = \mathsf{true}$ **and** $m^{\mathrm{mitm}} \neq \perp$, output $(\mathsf{Sent}, \mathsf{sid}, S, m^{\mathrm{mitm}})$ to R and set $\mathsf{delivered} \leftarrow \mathsf{true}$.

- Else if $\mathsf{auth} = \mathsf{false}$ (attempted and failed), **output nothing** and set $\mathsf{delivered} \leftarrow \mathsf{true}$.
- Else ($\mathsf{auth} = \perp$; no attempt), output $(\mathsf{Sent}, \mathsf{sid}, S, m)$ to R and set $\mathsf{delivered} \leftarrow \mathsf{true}$.

• **Ignore** all other inputs and back-door messages.

4 Universally Composable Authentication from SAS

This section puts SAS-based authentication on a UC footing. We expose a hybrid $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ channel for short, human-verifiable comparison and give two compact protocols, π_{SAS} and π_{SAS}^{X}, that follow a commit–respond–open pattern with a single SAS check. We prove they UC-realize $\mathcal{F}_{\mathrm{AUTH}}^{S \to R}(\varepsilon)$ and $\mathcal{F}_{\mathrm{AUTH}}^{S \leftrightarrow R}(\varepsilon)$ with $\varepsilon = 2^{-t}$ in the ROM.

4.1 Out-of-Band SAS Confirmation

One crucial component of our protocol is the authenticated *out-of-band channel* (OOB), which we model as a hybrid functionality and follow the modelling from [11]. Our abstraction captures the synchronous user interaction between the Sender and Receiver required to verify the SAS.

Ideal Functionality $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$

Parties: R (receiver) and S (sender) and the adversary $\mathcal{A}$.

• **SAS Comparison:** Upon input $(\mathsf{SAS}, \mathsf{sid}, s)$ from S or R, store the SAS and send $(\mathsf{SAS}, \mathsf{sid}, s)$ to $\mathcal{A}$. If the SAS from the other party was already received, i.e. s', compute the flag $b := (s == s')$ then return to the sender and receiver $(\mathsf{SAS}, \mathsf{sid}, b)$ (immediate delivery).

• **Ignore** all other inputs and back-door messages.

This channel differs from the usual model of data transmission over untrusted networks, where an adversary can arbitrarily drop messages. In our $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ hybrid, we assume the SAS is displayed to honest users on uncompromised hardware and software, so message delivery cannot be suppressed. Addressing a compromised display or denial-of-service attack on the OOB channel is an interesting direction for future work, but we leave it outside the scope of this paper. In the full version, we also discuss how human error and asynchronicity relate to our modeling choices.

4.2 Universally Composable SAS and OOB based One-Sided Authentication

We begin with a minimal, user-aided authentication primitive: a three-move SAS protocol that binds a sender's short context string to a receiver, using a single human comparison. The hash H is modeled as a random oracle and the displayed short authentication string (SAS) is the t-bit truncation of H on the session

transcript. We will argue that the adversary's best active success is $\varepsilon = 2^{-t}$, up to standard concurrency factors [34, 32, 28]. The human check is abstracted by the ideal functionality $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$, and we keep the network parties honest.

Real Protocol π_{SAS} (hybrid $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$)

Parties. S, R, the adversary $\mathcal{A}$, and hybrid $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$.
Parameters. Output length $t \in \mathbb{N}$. Let H be a random oracle and $\mathsf{SAS}(x) := \text{Trunc}_t(H(x))$.

0. Session setup (SID).
Either receive a session identifier sid from the calling environment, or (if not provided) have S sample a fresh $\mathsf{sid} \leftarrow\$ \{0,1\}^\kappa$ and notify R over the main channel. All local state and every call to subroutines/functionalities in this session is tagged with the same sid.

1. Commitment. $S \to R$: $(C, d) \leftarrow \mathsf{Commit}(\mathsf{sid} \parallel \mathsf{m}_S; \mathsf{r}_S)$, where r_S are the sender's random coins (contained in d); send C.

2. Receiver's Contribution. $R \to S$: sample $\mathsf{r}_R \leftarrow\$ \{0,1\}^\kappa$.

3. Reveal & SAS. $S \to R$: send $\mathsf{Open}(C, d)$.
If R rejects the opening, abort. Otherwise both compute

$$s \;:=\; \mathsf{SAS}\big(\mathsf{sid} \parallel \mathsf{m}_S \parallel \mathsf{r}_S \parallel \mathsf{r}_R\big)$$

and each party sends $(\mathsf{SAS}, \mathsf{sid}, s)$ to $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$.

4. Confirmation. $\mathcal{F}_{\text{OOB}}^{\text{SAS}} \to \{S, R\}$: $(\mathsf{SAS}, \mathsf{sid}, b)$.
R outputs m_S iff $b = 1$; otherwise abort.

Theorem 2 (UC security of π_{SAS} toward $\mathcal{F}_{\text{AUTH}}^{S\to R}(\varepsilon)$). *Let $t \in \mathbb{N}$ and define $\varepsilon := 2^{-t}$. Assume: (i) $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$ is computationally hiding and binding; (ii) H is modeled as a random oracle and $\mathsf{SAS}(x) = \text{Trunc}_t(H(x))$; (iii) the hybrid functionality $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ (SAS comparison with leakage to the adversary) is available. Then for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$,*

$$\mathsf{EXEC}^{\mathcal{F}_{\text{OOB}}^{\text{SAS}}}_{\mathcal{Z},\mathcal{A},\Pi_{\mathsf{SAS}}} \;\approx\; \mathsf{IDEAL}_{\mathcal{Z},\mathcal{S},\mathcal{F}_{\text{AUTH}}^{S\to R}(\varepsilon)}.$$

Hence, in the $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$-hybrid, Π_{SAS} UC-realizes the authenticated one-way channel with a single online active-forgery attempt that succeeds with probability ε.

Note that committing to the message, and thus hiding it in the first flow, is not strictly required. If the commitment scheme is more efficient when the committed values are small (and the message may be rather large), then it is perfectly fine to instead commit to some small random string and send the message bits in plaintext.

We nevertheless justify our choice by arguing that revealing the message in the last step is a more sensible approach: we follow the reveal step directly with

the out-of-band comparison and thus authenticate the message immediately. In contrast, revealing the (still unauthenticated) message in the first flow may leave some time before it is authenticated, since two network flows are still missing, and it might tempt security laypersons to pre-process the unauthenticated message.

The full proof can be found in Appendix A. Unfortunately, $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}$ is yet not sufficient for our requirements of a user-friendly protocol (albeit theoretically we can now effectively compose multiple instances and provide "virtual" bi-directional authentication), which is the same argument that [11] used in order to define the functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(k)$, which is a straight forward extension of $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}$ in order to allow bi-directional "back-and-forth" communication of k authenticate messages between S and R.

4.3 Universally Composable Cross-Authentication from SAS

In this section we extend π_{SAS} to support *bi-directional* delivery of one application message per party. The change is minimal: the receiver R now *piggybacks* its message m_R alongside its fresh nonce r_R in Step 2. The SAS then commits to the full transcript $(\mathsf{sid}, \mathsf{m}_S, \mathsf{m}_R, \mathsf{r}_S, \mathsf{r}_R)$, so any tampering that alters what either party sees requires a collision on the truncated random-oracle output. The session identifier sid is carried throughout to ensure subroutine-respecting composition. In the following we show that this change does not contradict the security.

Real Protocol π_{SAS}^{X} (hybrid $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$)

Parties. S, R, the adversary $\mathcal{A}$, and hybrid $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$.
Parameters. Output length $t \in \mathbb{N}$. Let H be a random oracle and $\mathsf{SAS}(x) := \mathrm{Trunc}_t(H(x))$. Let sid be the session identifier of this instance.

1. Commitment. $S \to R$: $(C, d) \leftarrow \mathsf{Commit}(\mathsf{sid} \parallel \mathsf{m}_S; \mathsf{r}_S)$, where r_S are the commitment coins (contained in d); send C.

2. Receiver's Contribution.

$R \to S$: choose m_R and sample $\mathsf{r}_R \leftarrow\$ \{0,1\}^{\kappa}$; send $(\mathsf{m}_R, \mathsf{r}_R)$.

3. Reveal & SAS. $S \to R$: send $\mathsf{Open}(C, d)$.
If R rejects the opening, abort. Else both compute

$$s \;:=\; \mathsf{SAS}\big(\mathsf{sid} \parallel \mathsf{m}_S \parallel \mathsf{m}_R \parallel \mathsf{r}_S \parallel \mathsf{r}_R\big)\,.$$

and send $(\mathsf{SAS}, \mathsf{sid}, s)$ to $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$.

4. Confirmation. $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}} \to \{S, R\}$: $(\mathsf{SAS}, \mathsf{sid}, b)$.
R outputs m_S and S outputs m_R iff $b = 1$; otherwise both abort.

Theorem 3 (UC security of π_{SAS}^{X} toward $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$). *Let $t \in \mathbb{N}$ and set $\varepsilon := 2^{-t}$. Assume: (i) $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$ is computationally hiding and*

binding; (ii) H is modeled as a random oracle and $\mathsf{SAS}(x) = \mathrm{Trunc}_t(H(x))$; (iii) the hybrid functionality $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ is available. Then for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$,

$$\mathsf{EXEC}^{\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}}_{\mathcal{Z},\,\mathcal{A},\,\pi^{X}_{\mathrm{SAS}}} \;\approx\; \mathsf{IDEAL}_{\mathcal{Z},\,\mathcal{S},\,\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)}.$$

Hence, in the $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$-hybrid, π^{X}_{SAS} UC-realizes cross-authentication $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ with a single online active-forgery attempt of success probability ε.

Proof. For the sake of brevity we only provide the most important differences to the previous proof.

- **SAS inputs (pair-aware).** At compare time use

$$x_R := (\mathsf{sid} \parallel \mathsf{m}'_S \parallel m_R \parallel \mathsf{r}'_S \parallel \mathsf{r}_R), \qquad x_S := (\mathsf{sid} \parallel \mathsf{m}_S \parallel \tilde{m}_R \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}}_R).$$

  Tampering is the semantic event $x_R \neq x_S$ (covers flips of m_R and/or r_R in transit in addition to commitment tamper).
- **Pre-hit hybrid.** $\mathsf{preHit}_{\mathsf{sid}} := [\, x_R \in T \vee x_S \in T \,]$ remains negligible; proof unchanged (requires guessing hidden r_S).
- **Tamper lemma.** Conditioned on binding and $x_R \neq x_S$, the RO outputs are independent uniform strings; hence $\Pr[\mathrm{Trunc}_t(H(x_R)) = \mathrm{Trunc}_t(H(x_S)) \mid x_R \neq x_S] = 2^{-t} \pm \mathsf{negl}(\kappa)$.
- **Coupling.** If $x_R = x_S$ (no tamper), emulate $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ with b=1 and, on $(\mathsf{ok}, \mathsf{sid})$, let $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}$ output the honest pair. If $x_R \neq x_S$ (tamper), query the forgery gate with $(\mathsf{Forge}, \mathsf{sid}, m'_S, m'_R)$ in $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$, get b, and prefix-program the RO so equality of the two t-bit SAS values holds iff b=1; return the same b from the emulated $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$.

Final distinguishing bound. Let $\Delta(\kappa)$ be the UC advantage between $\mathsf{EXEC}^{\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}}_{\mathcal{Z},\mathcal{A},\pi^{X}_{\mathrm{SAS}}}$ and $\mathsf{IDEAL}_{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)}$. Then

$$\Delta(\kappa) \;\le\; \mathcal{A}^{\mathrm{hide}}_{\mathsf{Com}}(\kappa) + \mathcal{A}^{\mathrm{bind}}_{\mathsf{Com}}(\kappa) + 2\, q_H(\kappa)\, 2^{-\kappa} + \Big|\, \Pr_{\mathrm{real}}[\mathrm{accept} \mid x_R \neq x_S] - \varepsilon \,\Big| \;=\; \mathsf{negl}(\kappa),$$

since $\Pr_{\mathrm{real}}[\mathrm{accept} \mid x_R \neq x_S] = 2^{-t} \pm \mathsf{negl}(\kappa)$ by the tamper lemma, and $\varepsilon = 2^{-t}$.

Final Simulator (differences vs. prior). The full simulator is included in the full version.

- **Ordering fix.** Matches the real-world message order $C \to (m_R, \mathsf{r}_R) \to$ reveal; the receiver's contribution is buffered until the commitment C is observed (no premature receiver start).
- **Other mechanisms unchanged.** Forge-gate invocation (single call on tamper, cached bit b), prefix-constrained lazy RO programming, and ok-coupled delivery remain as in the previous simulator.
- **Attempt-conditioned gate.** The $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}$ forge gate is invoked *iff* $x_R \neq x_S$.

□

5 Canonical Secure Message Transfer

We now move beyond mere authenticity to confidential and authenticated communication. That is, we aim to realize a standard secure–message–transmission functionality $\mathcal{F}_{\text{SMT}}^{S\to R}$ with the addition of a forge-gate attack vector, thus we will use $\mathcal{F}_{\text{SMT}}^{S\to R}(\varepsilon)$ from Fig. 3.4. In the previous section we proved that π_{SAS}^{X} UC-realizes $\mathcal{F}_{\text{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ (Theorem 3). Our first construction leverages π_{SAS}^{X} to authenticate public-key material (a KEM public key and a digital-signature verification key), after which payloads are sent via a KEM–DEM channel with signed ciphertexts. Our second construction minimizes asymmetric cryptography by replacing digital signatures with a MAC, whose key is contributed and bound through π_{SAS}^{X}. The second construction is presented in Appendix B. Our target in this section is the canonical realization of $\mathcal{F}_{\text{SMT}}^{S\to R}$ of Fig. 3.4. Following Canetti's recipe [14][11], we combine

- a KEM/DEM channel with a CPA-secure KEM and an OT-CPA-secure DEM for semantic confidentiality,
- an EUF-CMA digital signature to authenticate ciphertexts (and bind sender identity), and
- the cross-authentication functionality $\mathcal{F}_{\text{AUTH}}^{S\leftrightarrow R}$ (Section 3.4) to distribute the KEM public key and the signature verification key.
- a Session identifier sid to denote the SID of this protocol instance. We invoke the subroutine π_{SAS}^{X} with the same sid (subroutine-respecting).

and obtain the following protocol.

Real Protocol $\pi_{\text{SAS+KEM+DEM+Sig}}$ (hybrid $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$)

Primitives and model.

- **KEM** $(\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ IND-CPA.
- **DEM** $(\mathsf{ENC}, \mathsf{DEC})$ one-time IND-CPA (fresh key per session).
- **Sig** $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy})$ EUF-CMA.
- Cross-authentication subroutine π_{SAS}^{X} in the $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ hybrid.
- Signed context

$$\mathsf{ctx} \;:=\; \text{“}\pi_{\text{SAS+KEM+DEM+Sig}}\text{”} \,\|\, \mathsf{sid} \,\|\, \text{“S}\to\text{R”}\,.$$

Parties. Sender S and Receiver R.

Phase 1 — Cross-authentication via π_{SAS}^{X}.

1: S: $(\mathsf{vk}_S, \mathsf{sk}_S) \leftarrow \mathsf{Sig.KGen}$; set $m_S \leftarrow \mathsf{vk}_S$.
2: R: $(\mathsf{pk}_R, \mathsf{sk}_R) \leftarrow \mathsf{KEM.Gen}$; set $m_R \leftarrow \mathsf{pk}_R$.
3: Run π_{SAS}^{X} under the *same* sid on inputs (m_S, m_R).
4: Abort iff π_{SAS}^{X} outputs $b = 0$ (SAS mismatch).

Phase 2 — One-shot KEM–DEM (Enc-Then-Sign).

[11] See pp. 79 ff. and Claim 27 in the journal version.

(2a) Sender S with payload m:
1: $(K, c_{\mathsf{KEM}}) \leftarrow \mathsf{Encap}(\mathsf{pk}_R)$; $\quad c_{\mathsf{DEM}} \leftarrow \mathsf{DEM.ENC}(K, m)$
2: $\sigma \leftarrow \mathsf{Sig.Sign}_{\mathsf{sk}_S}\big(\mathsf{ctx}, c_{\mathsf{KEM}}, c_{\mathsf{DEM}}\big)$
3: $S \to R$: $(\mathsf{sid}, c_{\mathsf{KEM}}, c_{\mathsf{DEM}}, \sigma)$
(2b) Receiver R:
4: Abort if $\mathsf{Sig.Vfy}_{\mathsf{vk}_S}\big(\sigma; \mathsf{ctx}, c_{\mathsf{KEM}}, c_{\mathsf{DEM}}\big) = 0$
5: $K \leftarrow \mathsf{Decap}(\mathsf{sk}_R, c_{\mathsf{KEM}})$; $\quad m \leftarrow \mathsf{DEM.DEC}(K, c_{\mathsf{DEM}})$
6: Output $(\mathsf{Sent}, \mathsf{sid}, S, \mathsf{m})$ to the environment.

Theorem 4 (UC realization of $\mathcal{F}_{\mathrm{SMT}}^{S\to R}(\varepsilon)$ in the $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$-hybrid). *Assume:*

(i) KEM *is* IND-CPA*;*
(ii) DEM *is one-time* OT-CPA*;*
(iii) Sig *is* EUF-CMA*;*
(iv) π_{SAS}^{X} *UC-realizes* $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ *(one online misbinding with success ε).*

Protocol $\pi_{SAS+KEM+DEM+Sig}$ is subroutine-respecting (same sid *in π_{SAS}^{X} and in* ctx*) and uses fresh $(\mathsf{vk}_S, \mathsf{pk}_R)$ per session. Then, in the $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$-hybrid, $\pi_{SAS+KEM+DEM+Sig}$ UC-realizes $\mathcal{F}_{\mathrm{SMT}}^{S\to R}(\varepsilon)$ with length-only leakage. For any PPT environment,*

$$\mathcal{A}^{\mathrm{UC}} \;\le\; \varepsilon \;+\; \mathcal{A}_{\mathsf{Sig}}^{\mathsf{EUF\text{-}CMA}} \;+\; \mathcal{A}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CPA}} \;+\; \mathcal{A}_{\mathsf{DEM}}^{\mathsf{OT\text{-}CPA}} \;+\; \mathsf{negl}(\kappa)\,.$$

The full proof is presented in the full version.

6 Systematization of the Real-World Tooling Landscape

This section surveys how popular tools handle rendezvous, authentication, and transport when two endpoints want to "connect now" with minimal setup. Our goal is to extract the architectural patterns that matter for security and not to audit implementations. We read public docs and code as of 03 Sep 2025 and place each tool along a few common axes: (i) how peers meet (discovery/rendezvous), (ii) what—if anything—the human binds out of band (none / PAKE password / short SAS), (iii) which transport is used (and where encryption terminates), and (iv) what the back-end service learns under an active, malicious-server assumption.

We then look at two slices. First, section 6.1 systematizes one-shot file-transfer tools and contrasts self-signed DTLS data channels with PAKE-based "one code" designs, and with our SAS-based alternative. Second, section 6.2 reviews SAS-style checks in deployed systems and separates true *session-bound SAS* from longer, identity-fingerprint comparisons. Throughout, we connect these mechanisms to our modeling ($\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$, $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}$/$\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}$) to clarify what each pattern actually guarantees when the service operator is adversarial.

6.1 File Transfer in the Real World

We now narrow the scope to one-shot device–to–device transfers between co-located devices owned by the same person. We retain the adversarial-service threat model stated above: signaling/rendezvous infrastructure may be malicious, while on-the-wire transports provide confidentiality and integrity. Under this model the residual question is what the service learns and whether any human out-of-band bind (if any) defeats MitM. We begin with tools that form WebRTC data channels without a user-verifiable bind, then cover PAKE-based "one-code" designs, and finally contrast both with our SAS-based alternative. An asynchronous variant, where comparison occurs later or across users, entails different UX/trust trade-offs and is left for future work.

Unauthenticated DTLS. *PairDrop*[12], *ShareDrop*[13], *FilePizza*[14], and *Snapdrop*[15] establish WebRTC data channels without exposing any user-verifiable binding (no SAS, no Password, no fingerprint UI). Technically, browsers set up an SCTP association over DTLS over UDP. Each peer's self-signed DTLS certificate fingerprint is conveyed inside SDP via the signaling channel (see WebRTC security architecture RFC 8826/8827). If signaling is malicious, it can substitute SDPs and fingerprints and terminate two DTLS associations, achieving a full MitM despite DTLS on the wire. This is not a flaw in DTLS or SCTP but a missing *identity bind*: the browser or app verifies that the DTLS cert matches the fingerprint it received, but there are no additional checks that this fingerprint corresponds to the intended peer.

A clean mitigation is to add a user-verifiable bind. Our framework provides exactly that: a cross-authentication functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\leftrightarrow R}(\varepsilon)$ realized from a short-authentication-string (SAS) over an authenticated OOB channel. Concretely, one can display a short fingerprint/SAS derived from the agreed transcript and ask both sides to compare it over an auxiliary authenticated channel and if it matches, the session is user-bound and MitM is defeated under the stated $\varepsilon = 2^{-t}$ budget. The noisytransfer protocol implementation[16] and the companion `noisyauth` npm package[17] are available for this purpose. It is also worthwhile mentioning that at PairDrop it is already being discussed about adding similar protections[18].

PAKE-based. Tools like Magic Wormhole[19], WebWormhole[20], and croc[21] follow the same architectural pattern. The sender locally creates a single human-

[12] `https://github.com/schlagmichdoch/PairDrop`
[13] `https://github.com/ShareDropio/sharedrop`
[14] `https://github.com/kern/filepizza`
[15] `https://github.com/SnapDrop/snapdrop`
[16] `https://github.com/collapsinghierarchy/noisytransfer-protocol`
[17] `https://www.npmjs.com/package/@noisytransfer/noisyauth`
[18] `https://github.com/schlagmichdoch/PairDrop/issues/180`
[19] `https://github.com/magic-wormhole/magic-wormhole`
[20] `https://github.com/saljam/webwormhole`
[21] `https://github.com/schollz/croc`

readable code and shares it out of band with the receiver. Codes have usually the form `N-word-word...` (e.g., `7-crossover-clockwork`). That code combines two roles: a rendezvous reference (e.g., a mailbox nameplate or a server slot) and a PAKE password. The receiver inputs the code, which lets both sides attach to the same rendezvous and then run a PAKE to authenticate and derive keys. Two integration styles are common: (i) PAKE directly yields application-layer channel keys for encrypted streaming (e.g., SPAKE2 [24] in Magic Wormhole and croc), or (ii) PAKE authenticates[22] the signaling metadata before the data channel forms (e.g., CPace protects the SDP carrying DTLS certificate fingerprints in WebWormhole [20]). Under a malicious-server assumption, the service learns the rendezvous identifier and sees the PAKE transcript but not the password. Security then hinges on online-guess resistance and careful client-side handling. This pattern delivers a strong user experience: one code, typed once. It also defines a practical ceiling for PAKE-based UX because the out-of-band message must carry a sufficiently long shared secret alongside rendezvous data.

Our SAS-based approach decouples these concerns: use a short, non-secret rendezvous code purely for meeting, then display and compare a short SAS for binding. No long secret needs to be carried out of band, and the residual misbinding risk is explicit via the $\varepsilon = 2^{-t}$ budget, assuming a synchronous $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ is available as in our model. It is an interesting open question which of the paradigms provide the best usability/security tradefoff.

6.2 SAS Handling in the Real World

Beyond file transfer, many deployed systems ask users to compare short strings or longer fingerprints to detect MitM. The items below use *identity-key fingerprints* (often with a QR fallback), not a short, session-bound SAS in the formal sense. They are excellent for long-term identity verification in persistent conversations or ecosystems, but they do not authenticate a fresh one-shot transcript and their displayed values are much larger (e.g., 60 decimal digits) and change only when long-term keys rotate. Longer fingerprints are not primarily about limiting online guessing (as in SAS), but rather about making it infeasible to find alternative keys that yield the same displayed value. The relevant bound is given by the collision resistance (or, depending on the comparison model, second-preimage resistance) of the fingerprinting function.

From our understanding, a similar $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ could be used, for example, with a simple exchange of public-key material of a KEM and a digital signature, where the received public keys' fingerprints are compared. In that setting, however, there would be no human in the loop: the $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ would instead be a QR code that performs the comparison automatically. A proper treatment of this approach would require formal modeling of fingerprinting, which would greatly expand the scope of this work. We therefore exclude such approaches from our SAS-focused analysis, but we acknowledge that a formal analysis would be insightful and would enable rigorous comparisons.

[22] and unnecessarily encrypts

- **Signal.** Each 1:1 chat shows a safety number (numeric and QR) that fingerprints the parties' identity keys; when keys change, the app forces reverification [33]. This is an identity fingerprint, not a per-session SAS.
- **WhatsApp.** Each chat exposes a security code (60 digits and QR) that users compare/scan to verify the chat's long-term keys [35]; again a fingerprint rather than a live SAS.
- **Apple iMessage (Contact Key Verification).** Users compare short verification codes (or rely on posted public verification codes) to detect key substitution on Apple's servers; the check binds directory-backed identity keys, not a transient handshake [1, 2, 4].
- **Wire.** Conversations/devices expose key fingerprints (with QR) for out-of-band comparison, targeting identity/device trust establishment rather than per-run SAS binding [8, 3].
- **Threema.** Contacts are verified by scanning a QR code or comparing the public-key fingerprint to prevent MitM; this is a stable identity fingerprint, not recomputed per session [7, 6].
- **Telegram Secret Chats.** Clients display a key visualization derived from the chat's keys for users to compare; functionally this is a human-verifiable fingerprint for the chat, not a short SAS of a fresh run [5].

The following real-world examples constitute SAS-based authentication protocols for exchanged key material in various forms. We summarize the following analysis in Table 1.

ZRTP SAS derivation. ZRTP derives a human-verifiable SAS during its DH key agreement ([36], §4–§7).

- **Handshake setup.** After Hello/HelloAck ([36], §4–§7), the initiator A fixes its DH payload by sending Commit with a standard hash commitment $\mathsf{hvi} = \mathsf{Hash}(\mathsf{DHPart2}_A \,\|\, \mathsf{Hello}_B)$.
- **DH exchange and check.** Peers exchange DHPart1/DHPart2; verify $\mathsf{hvi} \stackrel{?}{=} \mathsf{Hash}(\mathsf{DHPart2}_A \,\|\, \mathsf{Hello}_B)$. Compute the shared DH value and derive the master secret s_0.
- **Context for SAS.** Set $\mathsf{KDF\_Context} := \mathsf{ZID}_i \,\|\, \mathsf{ZID}_r \,\|\, \mathsf{total\_hash}$, where ZID_* are the 96-bit ZRTP identifiers and total_hash is the running hash of all ZRTP messages in this run.
- **Derive SAS bits.** Using the spec's HMAC-based KDF (NIST SP 800-108 style; not necessarily HKDF), compute

$$\mathsf{sashash} := \mathsf{KDF}(s_0, \text{"}SAS\text{"}, \mathsf{KDF\_Context}, 256)$$

  and take $v := \mathrm{MSB}_{32}(\mathsf{sashash})$.
- **Render to users.** If B32 was negotiated, show the MSB 20 bits of v as four Base32 characters; if B256, show the MSB 16 bits as two PGP-word-list words ([36], §5.1.6). Users compare the display verbally/visually.

This is a classical SAS-derivation protocol in the Vaudenay/MANA sense: the commitment caps active attacker adaptivity, the SAS is bound to the live transcript via KDF_Context, and a successful MitM requires guessing the t-bit display. In our terminology, ZRTP aligns with π_{SAS}^{X} when we view the protocol messages as $m_S := \mathsf{DHPart2}_A$ and $m_R := \mathsf{DHPart1}_B$, and the verbal/visual code comparison as the $\mathcal{F}_{\text{OOB}}^{\text{SAS}}$ step. If the used KDF is modeled as a Random Oracle the universal composability from π_{SAS}^{X} thus also translates to ZRTP SAS-derivation.

Matrix/Element: Matrix implements a SAS verification (`m.sas.v1`) that follows the same Vaudenay/MANA commit, reveal, compare pattern as ZRTP: the acceptor first commits to its ephemeral key (hash over key and other information), both sides run ECDH, and the SAS is derived via HKDF-SHA256 from the shared secret and an info string binding user IDs, device IDs, both ephemeral keys, and a transaction id, then rendered as three 4-digit numbers or seven emoji [30]. After users confirm a match, Matrix additionally MAC-authenticates each device's long-term keys under a key from the same ECDH output, tying the human check to concrete device identities. At a high level this is the same session-bound SAS approach as ZRTP/MANA-IV/MA-DH. The small differences are the commitment target and the identity-binding step that follows the SAS.

Bluetooth LE Secure Connections (Numeric Comparison) SAS. In LE Secure Connections, both sides exchange ECDH public keys $(\text{PK}_A, \text{PK}_B)$, commit to fresh nonces via `Pairing Confirm`, reveal the nonces via `Pairing Random`, and then display a 6-digit value for human comparison (the "SAS"). This follows the MANA-IV commit, reveal and compare pattern, with the difference that both endpoints commit (each sends a confirm) rather than only the initiator. Concretely, the commitment values and the final SAS are:

$$\begin{aligned} C_A &= f4(\text{PK}_{Ax}, \text{PK}_{Bx}, N_A, 0) \\ C_B &= f4(\text{PK}_{Bx}, \text{PK}_{Ax}, N_B, 0) \\ \text{SAS} &= g2(\text{PK}_{Ax}, \text{PK}_{Bx}, N_A, N_B) \bmod 10^6 \end{aligned}$$

Where $f4$ and $g2$ are defined as AES-CMAC keyed by the (pre-reveal and thus still-secret) nonce N_A, with $g2$ first reduced $\bmod 2^{32}$ before rendering as six decimal digits. Security thus hinges on two CMAC-based components: (i) $f4$ as a computational "commitment" to the nonce bound to both public keys, and (ii) $g2$ as a PRF-derived short value over $(\text{PK}_{Ax}, \text{PK}_{Bx}, N_A, N_B)$. Under standard PRF/MAC assumptions for AES-CMAC and fresh, uniform nonces, $f4$ is *hiding* (the tag leaks nothing about the nonce prior to reveal) and *binding* (after sending C on a fixed input, opening to a different nonce would require finding a new CMAC key that reproduces C), and $g2$ yields a value computationally indistinguishable from uniform on $\{0,1\}^{32}$ (hence $\approx 10^{-6}$ success after decimal reduction).

However, unlike our π^{X}_{SAS} protocol, BLE's $g2$ is not modeled as a random oracle. Our UC simulator for π^{X}_{SAS} relies on a programmable random oracle to (A) guarantee exact uniformity under truncation and (B) program equal/different t-bit prefixes at the forge-gate. A fixed PRF such as CMAC does not provide that programmability. Thus, while BLE's Numeric Comparison is sound in the usual game-based sense, the UC simulation strategy we use for π^{X}_{SAS} does not carry over verbatim to BLE's SAS derivation. In the full version we also sketch an RO-free alternative (coin-flip c and commit-then-bind) that attains the same user interface while allowing a clean UC proof with standard UC commitments and a UC coin-flip ([15, 16, 12]). It is, however, conceptually different from the BLE handshake.

Protocol	Subsumed by our analysis?	Commitment primitive	SAS derivation primitive
π^{X}_{SAS} (this work)	Yes (reference UC-realizing SAS protocol).	Generic hiding and binding commitment $C = \mathsf{Commit}(\mathsf{sid} \parallel m_S; \mathsf{r}_S)$ with opening d (coins r_S).	$s := \mathrm{Trunc}_t\big(H(\mathsf{sid} \parallel m_S \parallel m_R \parallel \mathsf{r}_S \parallel \mathsf{r}_R)\big)$, where H is modelled as a random oracle and r_R is sampled by R.
ZRTP	Yes (under RO-style modeling of the KDF).	Hash-based commitment to the DH-transcript $\mathsf{Hash}(\mathsf{DHPart2}_A \parallel \mathsf{Hello}_B)$.	$\mathsf{KDF}(s_0, \text{`}sas\text{`}, \mathsf{context}, 256)$; SAS is the MSB t bits, rendered as Base32 / PGP words.
Matrix / Element (`m.sas.v1`)	Yes, up to HKDF modeling and the additional identity-MAC step.	Hash-based commitment to the acceptor's ephemeral key.	SAS from HKDF-SHA256 over the ECDH output and an info string; rendered as numbers or emoji.
BLE Secure Connections (Numeric Comparison)	Not directly: AES-CMAC-based SAS; our UC proof relies on RO programmability.	AES-CMAC tags $f4$.	$g2(\mathrm{PK}_{Ax}, \mathrm{PK}_{Bx}, N_A, N_B) \bmod 10^6$, where $g2$ is an AES-CMAC; displayed as six decimal digits;

Table 1. Session-bound SAS mechanisms and their relation to our UC analysis.

7 Conclusion

We studied instant, one-shot device–to–device exchanges under an active network adversary without PKI or pre-shared secrets. Our main result is that a short, human-verifiable comparison can be treated as a first-class resource with a precise, composable failure budget $\varepsilon = 2^{-t}$. We modeled this via $\mathcal{F}^{\mathrm{SAS}}_{\mathrm{OOB}}$, gave simple commitment-style protocols π_{SAS} and $\pi^{\times}_{\mathrm{SAS}}$ that are new variations of MANA IV, proved they realize $\mathcal{F}^{S\to R}_{\mathrm{AUTH}}(\varepsilon)$ and $\mathcal{F}^{S\leftrightarrow R}_{\mathrm{AUTH}}(\varepsilon)$, and showed how to compose them with standard KEM/DEM into $\mathcal{F}^{S\to R}_{\mathrm{SMT}}(\varepsilon)$ while preserving the explicit ε. Practically, this decouples rendezvous from authentication: the out-of-band step shrinks to comparing a short t-bit string instead of transporting a long shared secret.

Our systematization clarifies where today's tools stand. Browser WebRTC data channels commonly lack a user-verifiable bind and are thus vulnerable to adversarial signaling. PAKE "one-code" designs deliver strong usability but couple rendezvous codes with out-of-band secrets. Session-bound SAS fixes the missing bind with an explicit risk budget (identical to PAKEs) and can be retrofitted to existing stacks by hashing the negotiated transcript and showing a short code to both sides. We released reference code[23] to make this easy to integrate.

Limitations and future work. Our mainline proofs use a random-oracle instantiation for hashing and commitments. We sketched an RO-free variant based on a UC coin-flip and non-malleable commitments, but the full proof is left for future work and we need to rely upon a Common Reference String setup. We assumed synchronous, co-present users and set the human-error parameter $\varepsilon_{\mathrm{human}} = 0$ to isolate cryptographic guarantees. A calibrated treatment of human fallibility and asynchronous comparisons is also left to future work. Additional directions include group extensions, a stronger treatment of compromised UIs and denial-of-service (DoS) against the OOB channel, and analyzing SAS-based protocols within a deniability framework. Last but not least, extending our UC model to capture endpoint compromise and to provide some form of post-compromise security is outside the scope of this work and left for future research.

References

1. About imessage contact key verification. `https://support.apple.com/en-us/118246` (2023)
2. Advancing imessage security: Contact key verification. `https://security.apple.com/blog/imessage-contact-key-verification/` (2023)
3. How can i compare key fingerprints? `https://support.wire.com/hc/en-us/articles/207692235-How-can-I-compare-key-fingerprints` (2024)
4. Use contact key verification on iphone. `https://support.apple.com/guide/iphone/use-contact-key-verification-iph654dd8c53/ios` (2024)
5. Telegram secret chats: End-to-end encryption. `https://core.telegram.org/api/end-to-end` (2025), key visualization and fingerprint details
6. Threema support: Private faq (identity verification). `https://threema.com/en/support/private` (2025)
7. What is the qr code good for? `https://threema.com/en/faq/qr-code-expl` (2025)
8. Wire security & privacy. `https://wire.com/en/security` (2025)
9. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Annual international cryptology conference. pp. 232–249. Springer (1993)
10. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS. pp. 62–73 (1993). `https://doi.org/10.1145/168588.168596`
11. Benin, A., Toledo, S., Tromer, E.: Secure association for the internet of things. In: 2015 International Workshop on Secure Internet of Things (SIoT). pp. 25–34. IEEE (2015)

[23] `https://github.com/collapsinghierarchy/noisytransfer-protocol`

12. Blum, M.: Coin flipping by telephone. In: CRYPTO 1981 (reprinted in SIGACT News, 1983) (1983), commit-then-open coin tossing
13. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145 (oct 2001). https://doi.org/10.1109/SFCS.2001.959888
14. Canetti, R.: Universally composable security. Journal of the ACM (JACM) **67**(5), 1–94 (2020)
15. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO 2001. Springer (2001), uC commitments in CRS; enables UC coin-toss via commit⊕
16. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002 (2002), general UC framework; CRS-hybrid feasibility and building blocks
17. Cremers, C., Medinger, N., Naska, A.: Impossibility results for post-compromise security in real-world communication systems. In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 4391–4405. IEEE (2025)
18. Exchange, K.: Universally composable password-based. In: Advances in Cryptology-EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. vol. 3494, p. 404. Springer Science & Business Media (2005)
19. Goldberg, I., Mashatan, A., Stinson, D.R.: On message recognition protocols: Recoverability and explicit confirmation. International Journal of Applied Cryptography **2**(4), 331–346 (2010). https://doi.org/10.1504/IJACT.2010.038305, https://dl.acm.org/doi/10.1504/IJACT.2010.038305
20. Haase, B., Others: Cpace, a balanced composable pake. https://datatracker.ietf.org/doc/draft-irtf-cfrg-cpace/ (2025), internet-Draft, IRTF CFRG
21. Hammell, J., Weimerskirch, A., Girão, J., Westhoff, D.: Recognition in a low-power environment. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05) — Second International Workshop on Wireless Ad Hoc Networking (WWAN). pp. 933–938. IEEE Computer Society, Columbus, OH, USA (2005). https://doi.org/10.1109/ICDCSW.2005.119
22. Jarecki, S., Saxena, N.: Authenticated key agreement with key re-use in the short authenticated strings model. In: International Conference on Security and Cryptography for Networks. LNCS, vol. 6280, pp. 253–270. Springer, Springer (2010), https://nsaxena.engr.tamu.edu/wp-content/uploads/sites/238/2019/12/js-scn10.pdf
23. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, 3rd edn. (2020)
24. Ladd, W.: Spake2, a password-authenticated key exchange. https://www.rfc-editor.org/rfc/rfc9382 (September 2023), iRTF RFC 9382
25. Laur, S., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings. In: Cryptology and Network Security: 5th International Conference, CANS 2006, Suzhou, China, December 8-10, 2006, Proceedings. vol. 4301, pp. 90–107. Springer (2006)
26. Laur, S., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings (extended version). Tech. Rep. 2005/424, IACR ePrint (2006), https://kodu.ut.ee/~swen/publications/articles/laur-nyberg-2006.pdf
27. Laur, S., Pasini, S.: Sas-based group authentication and key agreement protocols. In: PKC 2008. LNCS, vol. 4939, pp. 197–213. Springer

(2008). https://doi.org/10.1007/978-3-540-78440-1_12, https://iacr.org/archive/pkc2008/49390198/49390198.pdf
28. Laur, S., Pasini, S.: User-aided data authentication. International Journal of Security and Networks **4**(1-2), 69–86 (2009)
29. Mashatan, A., Vaudenay, S.: A message recognition protocol based on standard assumptions. In: ACNS 2010. LNCS, vol. 6123, pp. 384–401. Springer (2010). https://doi.org/10.1007/978-3-642-13708-2_23, https://infoscience.epfl.ch/record/148699/files/MRP_ACNS10.pdf
30. Matrix.org: Implementing more advanced e2ee features (sas emoji/numbers). https://matrix.org/docs/guides/implementing-more-advanced-e-2-ee-features-such-as-cross-signing/ (2024), accessed 2025-09-03
31. Pasini, S., Vaudenay, S.: An optimal non-interactive message authentication protocol. In: CT-RSA 2006. LNCS, vol. 3860, pp. 280–296. Springer (2006). https://doi.org/10.1007/11605805_18, https://link.springer.com/chapter/10.1007/11605805_18
32. Pasini, S., Vaudenay, S.: Sas-based authenticated key agreement. In: International Workshop on Public Key Cryptography. pp. 395–409. Springer (2006)
33. Signal Messenger: What is a safety number and why do i see that it changed? https://support.signal.org/hc/en-us/articles/360007060632 (2024), accessed 2025-09-03
34. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 309–326. Springer-Verlag (2005)
35. WhatsApp: About security code change notifications. https://faq.whatsapp.com/1524220618005378 (2024), accessed 2025-09-03
36. Zimmermann, P.R., Johnston, A., Callas, J.: ZRTP: Media path key agreement for unicast secure RTP. RFC 6189 (2011), https://datatracker.ietf.org/doc/html/rfc6189

A Simulation Proof of $\pi_{\mathbf{SAS}}$

Theorem 5 (UC security of π_{SAS} toward $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$). *Let $t \in \mathbb{N}$ and define $\varepsilon := 2^{-t}$. Assume: (i) $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$ is computationally hiding and binding; (ii) H is modeled as a random oracle and $\mathsf{SAS}(x) = \mathrm{Trunc}_t(H(x))$; (iii) the hybrid functionality $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ (SAS comparison with leakage to the adversary) is available. Then for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$,*

$$\mathsf{EXEC}^{\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}}_{\mathcal{Z},\,\mathcal{A},\,\Pi_{\mathsf{SAS}}} \;\approx\; \mathsf{IDEAL}_{\mathcal{Z},\,\mathcal{S},\,\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)}.$$

Hence, in the $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$-hybrid, Π_{SAS} UC-realizes the authenticated one-way channel with a single online active-forgery attempt that succeeds with probability ε.

At a high level, in the ideal world the simulator receives the sender's input m_S from $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ via the back-door tape. Using this, it precomputes the honest transcript (including the t-bit SAS) and emulates the network and $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$

leakage exactly, preserving the real execution's scheduling. The only remaining distinguishing avenue is an active forgery/misbinding. Via a short hybrid sequence (commitment hiding/binding and RO unpredictability and programming) we bound its success by $\varepsilon = 2^{-t}$ and couple it to the Bernoulli(ε) gate of $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$.

hybrid argument. We work with the standard dummy-adversary reduction and indistinguishability of hybrids in the UC sense (cf. Canetti [14]). We keep S, R honest throughout; the adversary controls network scheduling and sees the $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ leakage.

Notation. Fix a session identifier sid chosen by the environment. The sender's payload is m_S; the receiver samples $\mathsf{r}_R \leftarrow\$ \{0,1\}^\kappa$; the sender's commitment coins are $\mathsf{r}_S \leftarrow\$ \{0,1\}^\kappa$. We model H as a random oracle with output length $n = n(\kappa)$ and write $\mathrm{Trunc}_t : \{0,1\}^n \to \{0,1\}^t$. Unless stated otherwise,

$$s \;:=\; \mathrm{Trunc}_t\big(H(\mathsf{sid} \parallel m_S \parallel \mathsf{r}_S \parallel \mathsf{r}_R)\big).$$

Random-oracle table. We implement H via *lazy sampling* using a global partial map

$$T : \{0,1\}^* \rightharpoonup \{0,1\}^n,$$

initially empty and shared across all ITMs in the execution (parties, adversary, and simulator). On a query x to H: if $x \in \mathrm{dom}(T)$ return $T[x]$; otherwise sample $y \leftarrow\$ \{0,1\}^n$ uniformly, set $T[x] \leftarrow y$, and return y. We say "*program T at x*" to mean assigning $T[x]$ to some value (possibly under a t-bit prefix constraint in the coupling step), always without overwriting existing entries. Throughout, $x \in T$ abbreviates $x \in \mathrm{dom}(T)$.

For the SAS comparison we will form

$$x_R := (\mathsf{sid} \parallel \mathsf{m}'_S \parallel \mathsf{r}'_S \parallel \mathsf{r}_R) \quad \text{and} \quad x_S := (\mathsf{sid} \parallel \mathsf{m}_S \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}}_R),$$

taken from the values actually seen on R's and S's wires, respectively. We use the event $\mathsf{preHit}_{\mathsf{sid}} := [\, x_R \in T \;\vee\; x_S \in T \,]$ to denote a *pre-reveal RO hit* in session sid. The commitment's decommitment is denoted d.

Game H0 (Real). The real execution of Π_{SAS} in the $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$-hybrid against $\mathcal{A}$.

H1 (Code lifting). We introduce an intermediate wrapper that internally runs *all* honest parties and the hybrid $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$, and forwards wires to $\mathcal{A}$ and $\mathcal{Z}$ verbatim. This is the usual "all-powerful simulator shell" that only reorganizes code, not distributions.

Lemma 1. $\mathsf{EXEC}[\mathrm{H0}] \equiv \mathsf{EXEC}[\mathrm{H1}]$ *(perfectly indistinguishable).*

H2 (Hiding-based replacement of the commitment). In H2, the wrapper replaces the real $C \leftarrow \mathsf{Commit}(\mathsf{sid} \parallel m_S; \mathsf{r}_S)$ by a uniform string $C \leftarrow\$ \{0,1\}^{n(\kappa)}$, while still opening it later to $(\mathsf{sid} \parallel m_S, d)$ as in the real protocol (i.e., it stores a table to ensure a consistent decommitment). Collisions among sampled C are ruled out with overwhelming probability via the birthday bound.

Lemma 2 (Commitment hiding). *If* Com *is computationally hiding, then* $\mathsf{EXEC}[\mathrm{H1}] \approx \mathsf{EXEC}[\mathrm{H2}]$.

Sketch. A standard reduction: a distinguisher between H1 and H2 yields a PPT adversary that breaks hiding by telling apart a commitment to m_S from a random string placed on the same wire. □

H3 (Abort on BAD double-opening). From H2, define H3 that aborts and outputs a special symbol Bad if the adversary ever induces a valid *second* opening $(m', d') \neq (m_S, d)$ for the same C.

Lemma 3 (Commitment binding). *If* Com *is binding then* $\Pr[\mathsf{Bad}]$ *is negligible, hence* $\mathsf{EXEC}[\mathrm{H2}] \approx \mathsf{EXEC}[\mathrm{H3}]$.

H4^{pre} (Flag pre-reveal RO hits). $\mathrm{H4}^{\mathrm{pre}}$ is identical to H3 except it sets a boolean flag $\mathsf{preHit}_{\mathsf{sid}} \leftarrow \mathsf{true}$ if, *before* the SAS comparison in session sid, the adversary has queried the RO at either x_R or x_S (as defined later in H4), i.e., if $x_R \in T$ or $x_S \in T$ holds at SAS time. Execution proceeds unchanged.

Lemma 4 (Pre-reveal RO hits are negligible). *Let* $\mathsf{r}_S \leftarrow\$ \{0,1\}^{\kappa}$ *be sampled and hidden inside the sender's commitment (so* $x_R := (\mathsf{sid} \parallel m'_S \parallel \mathsf{r}'_S \parallel \mathsf{r}_R)$ *and* $x_S := (\mathsf{sid} \parallel m_S \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}}_R)$ *contain* r_S*). Then for any PPT adversary making* $q_H(\kappa)$ *RO queries,*

$$\Pr[\mathsf{preHit}_{\mathsf{sid}}] \;\le\; 2\, q_H(\kappa) \cdot 2^{-\kappa} \;=\; \mathsf{negl}(\kappa).$$

Consequently, $\mathsf{EXEC}[\mathrm{H3}] \approx \mathsf{EXEC}[\mathrm{H4}^{\mathrm{pre}}]$.

Proof. Before reveal, (m_S, r_S) is hidden; in particular r_S is uniform and unknown. Hitting x_R (or x_S) with an RO query requires guessing r_S, which succeeds with probability at most $q_H \cdot 2^{-\kappa}$ per input. A union bound over $\{x_R, x_S\}$ gives the claim. □

H4 (Flag tamper and identical-until-bad). H4 is identical to H3 except it sets a boolean flag $\mathsf{tampered}_{\mathsf{sid}} \leftarrow \mathsf{true}$ if, in session sid, the adversary alters either (i) the sender's first message (a valid commitment C' is placed on $S{\to}R$ that differs from the honest C), or (ii) the receiver's nonce in flight ($\tilde{\mathsf{r}}_R \neq \mathsf{r}_R$ is placed on $R{\to}S$). At the SAS comparison, define

$$s_R \;:=\; \mathrm{Trunc}_t\big(H(\mathsf{sid} \parallel \mathsf{m}'_S \parallel \mathsf{r}'_S \parallel \mathsf{r}_R)\big), \qquad s_S \;:=\; \mathrm{Trunc}_t\big(H(\mathsf{sid} \parallel \mathsf{m}_S \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}_R})\big),$$

where in the non-tampered case $\tilde{\mathsf{r}}_R = \mathsf{r}_R$, $\mathsf{m}'_S = \mathsf{m}_S$ and $\mathsf{r}'_S = \mathsf{r}_S$. Let the *bad* event be $\mathsf{bad}_{\mathsf{sid}} := [\,\mathsf{tampered}_{\mathsf{sid}} \wedge (s_R = s_S)\,]$. Execution proceeds unchanged (no abort); we only *record* $\mathsf{bad}_{\mathsf{sid}}$.

Lemma 5 (SAS equality under tampering). *Model H as a random oracle. Conditioned on* $\mathsf{tampered}_{\mathsf{sid}}$ *and on the commitment being binding to m_S, the two inputs* $(\mathsf{sid} \parallel \mathsf{m}'_S \parallel \mathsf{r}'_S \parallel \mathsf{r}_R)$ *and* $(\mathsf{sid} \parallel \mathsf{m}_S \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}}_R)$ *are distinct with overwhelming probability; hence s_R and s_S are independent uniform t-bit strings. Therefore,* $\Pr[\mathsf{bad}_{\mathsf{sid}}] \leq 2^{-t} + \mathsf{negl}(\kappa)$*, and by the fundamental lemma of game-playing,*

$$\left| \Pr[\mathsf{EXEC}[\mathrm{H3}] \in \mathcal{V}] - \Pr[\mathsf{EXEC}[\mathrm{H4}] \in \mathcal{V}] \right| \;\leq\; \Pr[\mathsf{bad}_{\mathsf{sid}}] \;\leq\; 2^{-t} + \mathsf{negl}(\kappa)$$

for any PPT test $\mathcal{V}$.

Proof. Binding fixes m_S (and r_S if included) from R's perspective once C is opened honestly. If the adversary injects a different commitment C', it must also provide a valid opening to make R proceed; except with negligible probability this opening yields $(\mathsf{m}'_S, \mathsf{r}'_S) \neq (\mathsf{m}_S, \mathsf{r}_S)$, thereby diverging from S's view. The only remaining effective tamper point is delivering $\tilde{\mathsf{r}}_R \neq \mathsf{r}_R$ to S, which then diverges from R's view that uses r_R.

In the random-oracle model, distinct inputs yield independent uniform outputs; after truncation, $\Pr[s_R = s_S] = 2^{-t}$. Any residual probability mass is due only to violating binding or forcing an RO collision on distinct inputs, both of which are negligible in κ. □

H5 (Coupling to the ideal Bernoulli gate and Prefix-constrained lazy RO). We define a simulator $\mathcal{S}$ that, on top of H4's code lifting, replaces the real network and $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ by a perfect emulation driven by the ideal functionality $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$, while maintaining a lazy-sampled RO table T.

- **On** $(\mathsf{Sent}, \mathsf{sid}, S, \mathsf{m}_S)$ **from** $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$**:** $\mathcal{S}$ starts a local simulation of the real transcript: sample $C \leftarrow\$ \{0,1\}^{n(\kappa)}$, $\mathsf{r}_R \leftarrow\$ \{0,1\}^{\kappa}$ (and any sender coins embedded in the commitment), relay C on $S{\to}R$, then (under adversarial scheduling) relay r_R on $R{\to}S$, then the opening $(\mathsf{m}_S, \mathsf{r}_S)$ on $S{\to}R$. No accept bit is returned yet; $\mathcal{S}$ only mirrors $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$'s leakage interface later at SAS time.
- **At the SAS comparison (semantic tamper test):** Form

  $$x_R := (\mathsf{m}'_S \parallel \mathsf{r}'_S \parallel \mathsf{r}_R), \qquad x_S := (\mathsf{m}_S \parallel \mathsf{r}_S \parallel \tilde{\mathsf{r}}_R),$$

  where x_R uses the actually received opening on R's side and x_S uses whatever $\tilde{\mathsf{r}}_R$ arrived at S.
  - *If $x_R = x_S$ (no tamper):* program T lazily at x_R (if needed), let both parties' SAS be $s := \mathrm{Trunc}_t(H(x_R))$, leak $(\mathsf{SAS}, \mathsf{sid}, s)$ to $\mathcal{A}$ for each party, and return $b{=}1$ from the emulated $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$. Later, upon $(\mathsf{ok}, \mathsf{sid})$ from $\mathcal{A}$, forward it to $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ so that it delivers $(\mathsf{Sent}, \mathsf{sid}, S, \mathsf{m}_S)$ to R.
  - *If $x_R \neq x_S$ (tamper):* invoke the ideal gate by sending $(\mathsf{Forge}, \mathsf{sid}, \mathsf{m}', R)$ to $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ (where m' is the message determined by the adversarial flow) and receive $(\mathsf{ForgeResult}, \mathsf{sid}, b)$. Now program the RO lazily with a t-bit prefix constraint consistent with T:

- ∗ If neither x_R nor x_S is in T: sample $y_R, y_S \leftarrow\$ \{0,1\}^n$ uniformly subject to $\mathrm{Trunc}_t(y_R) = \mathrm{Trunc}_t(y_S)$ iff b=1; set $T[x_R] \leftarrow y_R$, $T[x_S] \leftarrow y_S$.
- ∗ If exactly one is in T: sample the other uniformly subject to the same t-bit relation to the existing one and add it to T.
- ∗ If both are already in T: if the existing t-bit relation contradicts b, this is precisely the negligible pre-reveal hit isolated in Lemma 4; otherwise proceed.

Let $s_R = \mathrm{Trunc}_t(T[x_R])$, $s_S = \mathrm{Trunc}_t(T[x_S])$; leak $(\mathsf{SAS}, \mathsf{sid}, s_R)$ and $(\mathsf{SAS}, \mathsf{sid}, s_S)$ to $\mathcal{A}$ as $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ would, and return that same bit b from the emulated $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ to both parties.

– **When $\mathcal{A}$ later sends** $(\mathsf{ok}, \mathsf{sid})$**:** If no successful forgery occurred (i.e., $b \neq 1$ in the tampered case), forward $(\mathsf{ok}, \mathsf{sid})$ to $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ to deliver $(\mathsf{Sent}, \mathsf{sid}, S, \mathsf{m}_S)$ to R.

Crucially, in the real world, any successful misbinding with honest parties requires that the two t-bit SAS values coincide on *distinct* RO inputs. By Lemma 5, this occurs with probability at most $\varepsilon + \mathsf{negl}(\kappa)$, where $\varepsilon = 2^{-t}$. The coupling above replaces this event by the Bernoulli(ε) gate of $\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)$ and mirrors the same accept/reject outcome in the emulated $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$, while preserving scheduling and leakage.

Lemma 6. $\mathsf{EXEC}[\mathrm{H5}] \approx \mathsf{IDEAL}[\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon), \mathcal{S}]$.

Conclusion. By transitivity of indistinguishability across H0–H5, we obtain the claim of Theorem 5.

Final distinguishing bound. Let $\Delta(\kappa)$ denote the UC distinguishing advantage of any PPT $(\mathcal{A}, \mathcal{Z})$ pair between $\mathsf{EXEC}^{\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}}_{\mathcal{Z},\mathcal{A},\Pi_{\mathsf{SAS}}}$ and $\mathsf{IDEAL}_{\mathcal{Z},\mathcal{S},\mathcal{F}_{\mathrm{AUTH}}^{S\to R}(\varepsilon)}$. Then

$$\Delta(\kappa) \;\le\; \underbrace{\mathcal{A}^{\mathrm{hide}}_{\mathsf{Com}}(\kappa)}_{\mathrm{H1}\to\mathrm{H2}} + \underbrace{\mathcal{A}^{\mathrm{bind}}_{\mathsf{Com}}(\kappa)}_{\mathrm{H2}\to\mathrm{H3}} + \underbrace{\Pr[\mathsf{preHit}_{\mathsf{sid}}]}_{\le\, 2\, q_H(\kappa)\, 2^{-\kappa}} + \underbrace{\Big|\, \Pr_{\mathrm{real}}[\mathrm{accept} \mid x_R \neq x_S] \;-\; \varepsilon \,\Big|}_{\le\, \mathsf{negl}(\kappa)\ \text{by Lemma 5},\ \varepsilon = 2^{-t}}$$

and thus

$$\Delta(\kappa) \;\le\; \mathcal{A}^{\mathrm{hide}}_{\mathsf{Com}}(\kappa) \;+\; \mathcal{A}^{\mathrm{bind}}_{\mathsf{Com}}(\kappa) \;+\; 2\, q_H(\kappa)\, 2^{-\kappa} \;+\; \mathsf{negl}(\kappa) \;=\; \mathsf{negl}(\kappa).$$

The full simulator is given in the full version. □

Attempt-conditioned gate. We sample the Bernoulli gate with parameter $\varepsilon = 2^{-t}$ *iff* the run reaches a genuine tamper point at SAS time, namely $x_R \neq x_S$. Let $I \in \{0,1\}$ indicate this event. In the real world, conditioned on I=1 the two RO inputs are distinct, so the accept probability is $\varepsilon \pm \mathsf{negl}(\kappa)$; hence

$$\Pr[\mathsf{accept}]_{\mathrm{real}} \;=\; \Pr[I{=}1] \cdot \varepsilon \;\pm\; \mathsf{negl}(\kappa).$$

In the ideal world, the simulator triggers the gate exactly when $I{=}1$, and the functionality returns $\mathrm{Ber}(\varepsilon)$, yielding

$$\Pr[\mathsf{accept}]_{\mathrm{ideal}} \;=\; \Pr[I{=}1] \cdot \varepsilon.$$

Thus an adversary who "plays weak" (e.g., tampers rarely or avoids creating $x_R \neq x_S$) merely decreases $\Pr[I{=}1]$ in *both* worlds; it cannot alter the per-attempt success, which is fixed at ε. Consequently the overall success probabilities match up to the negligible terms already isolated in the hybrids (pre-hit and RO-collision events).

B MAC-based SMT

Our next goal is to reduce the asymmetric footprint of the SMT stack by replacing the signature layer with symmetric authentication, while keeping the SAS-based cross-authentication as the only out-of-band (OOB) trust anchor. The main motivation is efficiency: asymmetric primitives are typically more costly than symmetric ones, as argued in [11]. This follows a common design line in SAS/PAKE-backed channels, where public-key signing can be traded for MACs keyed by ephemeral session material, but at the price of losing public verifiability and the "signature firewall" (i.e., verification before any decryption/long-term use). We note that the loss of public verifiability may, in some settings, support deniability goals, however a formal deniability analysis of our construction is out of scope. We retain the UC modelling of SAS via $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ and cross-authentication via π_{SAS}^{X}, and we continue to target the same ideal functionality $\mathcal{F}_{\mathrm{SMT}}^{S\to R}(\varepsilon)$ with length-only leakage and a single online misbinding event of success probability $\varepsilon = 2^{-t}$.

Real Protocol $\pi_{\mathbf{SAS+KEM+DEM+MAC}}$ (hybrid $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$)

Primitives.

- Equivocable commitment $\mathsf{Com} = (\mathsf{Commit}, \mathsf{Open})$: hiding, binding, with trapdoor for equivocation (used only by the UC simulator).
- KEM $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$: IND-CPA.
- DEM $\mathsf{DEM} = (\mathsf{ENC}, \mathsf{DEC})$: *one-time* IND-CPA (fresh KEM key per session).
- MAC (e.g. HMAC) with UF-CMA security.
- Random oracle H for SAS; $\mathrm{sas}(x) = \mathsf{Trunc}_t(H(x))$.

Parties. Sender S sends a single message m to receiver R.

Context string.

$$\mathsf{ctx} \;:=\; \text{“}\pi_{\mathrm{SAS+KEM+DEM+MAC}}\text{”} \,\|\, \mathsf{sid} \,\|\, \text{“S}\to\text{R”}\,.$$

Phase 1 — Cross-authentication via π_{SAS}^{X} (SID reuse)

1: S: sample $K_{\mathsf{MAC}} \leftarrow\$\ \{0,1\}^{\kappa}$ and commitment coins r_S; set $C_S \leftarrow \mathsf{Commit}(K_{\mathsf{MAC}}; r_S)$; define $m_S \leftarrow C_S$.

2: R: $(\mathsf{pk}_R, \mathsf{sk}_R) \leftarrow \mathsf{KEM.Gen}$; sample $\mathsf{r}_R \leftarrow\$ \{0,1\}^\kappa$; define $m_R \leftarrow \mathsf{pk}_R$.
3: *Run* π_{SAS}^X under the *same* sid with inputs (m_S, m_R).
4: **Abort** iff π_{SAS}^X outputs $(\mathsf{SAS}, \mathsf{sid}, 0)$.
5: *(On success b=1: S learns $(\mathsf{pk}_R, \mathsf{r}_R)$ and R learns C_S; SAS leakage flows via $\mathcal{F}_{\mathrm{OOB}}^{\mathrm{SAS}}$ as usual.)*

Phase 2 — One-shot transport with post-decrypt MAC check

(2a) Sender S with payload m:
1: $(K, c_{\mathsf{KEM}}) \leftarrow \mathsf{KEM.Encap}(\mathsf{pk}_R)$
2: $P \leftarrow \big(m,\, K_{\mathsf{MAC}},\, r_S\big)$ *(carry MAC key & its opening inside the DEM payload)*
3: $c_{\mathsf{DEM}} \leftarrow \mathsf{DEM.ENC}(K, P)$
4: $\tau \leftarrow \mathsf{MAC}_{K_{\mathsf{MAC}}}\big(\mathsf{ctx} \,\|\, c_{\mathsf{KEM}} \,\|\, c_{\mathsf{DEM}}\big)$
5: $S \to R : (\mathsf{sid}, c_{\mathsf{KEM}}, c_{\mathsf{DEM}}, \tau)$
(2b) Receiver R:
6: $K \leftarrow \mathsf{KEM.Decap}(\mathsf{sk}_R, c_{\mathsf{KEM}})$ *(abort on failure)*
7: $P \leftarrow \mathsf{DEM.DEC}(K, c_{\mathsf{DEM}})$; parse P as $(m, K_{\mathsf{MAC}}, r_S)$ *(abort on failure)*
8: **Commitment check:** accept only if $\mathsf{Open}(C_S; r_S, K_{\mathsf{MAC}})$ succeeds
9: **Tag check:** accept only if $\mathsf{MAC}_{K_{\mathsf{MAC}}}\big(\mathsf{ctx} \,\|\, c_{\mathsf{KEM}} \,\|\, c_{\mathsf{DEM}}\big) = \tau$
10: If both checks pass, output $(\mathsf{Sent}, \mathsf{sid}, S, m)$ to the environment; else abort.