

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Crypto Wars in Secure Messaging: Covert Channels in Signal Despite Leaked Keys

Rosario Giustolisi, *IT University of Copenhagen;* Gabriele Lenzini, *University of Luxembourg;* Chuanwei Lin, *University of Luxembourg and Karlsruhe Institute of Technology;* Mohammadamin Rakeei, *University of Luxembourg;* Andy Rupp, *University of Luxembourg and KASTEL Security Research Labs*

https://www.usenix.org/conference/usenixsecurity26/presentation/giustolisi

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

Crypto Wars in Secure Messaging: Covert Channels in Signal Despite Leaked Keys*

Rosario Giustolisi
IT University of Copenhagen

Gabriele Lenzini
University of Luxembourg

Chuanwei Lin
University of Luxembourg
Karlsruhe Institute of Technology

Mohammadamin Rakeei
University of Luxembourg

Andy Rupp
University of Luxembourg
KASTEL Security Research Labs

Abstract

End-to-end encryption (E2EE) is the foundation of modern secure messaging, with the Signal protocol as the de facto standard in applications such as Signal, WhatsApp, Facebook Messenger and Google Messages. At the same time, the deployment of E2EE has led to growing pressure from authorities to decrypt user traffic under law enforcement. This raises a critical question: if an adversary can routinely decrypt Signal messages (for example via a mandated access or a leaked key), can users still communicate securely and covertly?

We address this question through the lens of anamorphic encryption, which enables hidden communication within seemingly legitimate ciphertexts, even against an adversary who can decrypt them. We design two constructions that embed covert channels into the existing Signal Double Ratchet protocol. Concretely, we show how to embed covert messages (i) into Diffie-Hellman keys used in the asymmetric ratchet, or (ii) into authentication tags produced in the symmetric ratchet. Our techniques are compatible with existing Signal-style deployments and require no changes by the service provider.

We formalize security in threat models that capture adversaries with decryption capabilities granted through lawful-access mechanisms, and prove that the resulting protocol transcripts are indistinguishable from those of standard Signal. We implement our constructions in the official Signal library and Android client, and show that they incur low overhead and are practical in real-world settings. Our results show that covert communication channels can persist even when conventional E2EE guarantees are compromised.

1 Introduction

End-to-end encryption (E2EE) has become essential infrastructure for private communication. The Signal protocol [34, 36] is the foundation of messaging for billions of users across Signal, WhatsApp, Facebook Messenger, and Google Messages. Its Double Ratchet algorithm provides forward secrecy and post-compromise security, properties that hold up under extensive formal analysis [1, 7, 9, 15, 16, 29, 47] and that users increasingly rely on for secure communication.

At the same time, E2EE is under growing political pressure. Proposals in the EU [22], the UK [46], Sweden [18], and Australia [17] aim to require providers to enable decryption of E2EE traffic, through key escrow, client-side scanning, or other technical mandates. Signal has stated it would leave certain markets rather than comply [17, 18]. Other providers may comply, whether by choice or under legal pressure. For users who depend on E2EE, including journalists and sources, activists and organizers, lawyers and clients, and people in abusive situations, a practical question follows: if an adversary can decrypt your messages, is there still anything you can communicate privately?

We explore this question using *anamorphic encryption* [37]. Informally, anamorphic encryption lets a sender embed a second, covert message into a normal-looking ciphertext. An adversary who holds the regular decryption key can decrypt the visible message, yet cannot distinguish anamorphic ciphertexts from ordinary ones. Only someone with a separate *double key* can recover the hidden content. This contrasts with cover-media steganography, which hides data in objects such as images or audio and depends on access to suitable cover distributions. In practice, cover availability and robustness can be a bottleneck under sanitization, filtering, or censorship [32]. Anamorphic encryption sidesteps these constraints by operating inside cryptographic protocols. The adversary may decrypt the overt communication and inspect the resulting protocol values, so the covert channel must remain hidden under cryptographic inspection rather than rely on external cover objects. Prior work has developed the theory of anamorphic encryption [6, 12, 13, 32, 37, 49], but it has not been instantiated inside a widely deployed messaging protocol.

Applying anamorphic encryption to Signal raises protocol-specific challenges. Signal is not a single encryption call. The Double Ratchet evolves state over time, mixes asymmetric and symmetric steps, and must stay correct under message loss and

*Authors are listed alphabetically by last name. Mohammadamin Rakeei and Chuanwei Lin contributed equally.

out-of-order delivery. Protocol messages also carry multiple cryptographic values, such as DH public keys, ciphertexts, and authentication tags. A covert channel must blend into this full transcript, not just into one isolated primitive. It must also match the surveillance setting, since different disclosure patterns may reveal different pieces of key material.

We show that anamorphic communication can be realized within the Signal Double Ratchet with provable security and practical performance. We build two covert channels that hide data in different protocol components: one embeds covert messages in Diffie-Hellman (DH) public keys, and the other embeds covert messages in MAC authentication tags. In both cases, the resulting transcripts are indistinguishable from standard Signal executions for an adversary that can decrypt ordinary messages under the relevant key-disclosure assumptions. Empirically, the MAC-based channel incurs negligible overhead, and the optimized DH-based channel adds only a few milliseconds of overhead for short covert messages. Together, we show that covert communication can persist inside widely deployed E2EE protocols even when their primary confidentiality guarantees are weakened.

1.1 Contributions

We design, analyze, and implement covert channels for the Signal Double Ratchet. Our contributions are as follows.

Two practical ways to embed covert messages into Signal. We propose two families of constructions that hide data in different parts of the protocol and therefore apply under different key-disclosure patterns.

- **DH-based channel.** We embed covert messages into the Diffie-Hellman public keys generated during the asymmetric ratchet. The construction produces public keys that are computationally indistinguishable from honestly generated ones, even to an adversary who learns the Double Ratchet secrets that suffice to decrypt ordinary messages. We provide a symmetric double-key variant that requires pre-sharing the double key, and an asymmetric variant that allows covert initiation without prior coordination, assuming the receiver has published a corresponding public key.
- **MAC-based channel.** We embed covert messages into authentication tags in the symmetric ratchet. This construction requires no pre-shared double key, and applies in settings where surveillance yields only partial state. In particular, if the adversary learns either the root key or the DH private keys, but not both, then the derived MAC keys remain unknown to the adversary and can carry covert messages.

To our knowledge, these are the first anamorphic constructions realized within a deployed E2EE protocol.

Security definitions and proofs. We formalize *anamorphic security* via transcript indistinguishability: an adversary with access to the disclosed key material cannot distinguish anamorphic executions from standard Signal executions. We prove that our constructions satisfy this notion under standard cryptographic assumptions. We further extend both constructions with transcript binding to obtain IND-CCA2-style security against active adversaries who may inject, replay, or modify messages.

Concrete instantiation, optimization, and evaluation. We provide concrete instantiations and optimizations, and validate them in an end-to-end implementation.

Our DH-based channel is instantiated over X25519 using Edwards-Montgomery mappings. On the receiver side, naively extracting a covert message would require an exhaustive search over the message space, which quickly becomes impractical. We therefore introduce a lookup-table optimization that replaces exhaustive search with a single group operation followed by a table lookup, making a 32-bit covert message per DH ratchet step practical. The MAC-based channel supports messages up to the MAC length, which is 64 bits in our implementation.

We implement our constructions in a Signal-compatible Double Ratchet library and integrate them into an Android client, then evaluate their performance. In our measurements, the MAC-based channel incurs negligible overhead compared to unmodified Signal, while the optimized DH-based channel avoids the otherwise high extraction latency and achieves practical overhead for short covert messages. Both designs require no Signal server-side changes and remain compatible with existing Signal-style deployments. We also examine timing side channels, since decryption latency can leak whether a message carries covert content, and we discuss mitigations toward constant-time behavior.

1.2 Threat Model and Applicability

We consider an adversary who obtains Double Ratchet key material through a surveillance mechanism and can decrypt ordinary Signal messages. Different settings expose different subsets of the ratchet state, and our constructions are tailored to these disclosure patterns.

Full state disclosure. In the strongest setting, the adversary learns the complete Double Ratchet state used to decrypt and authenticate messages, including the root key, DH private keys, chain keys, and derived message keys. This can arise from broad key-disclosure requirements (as contemplated by recent EU and UK proposals [22, 46]), from taking a device and recovering secrets through forensic analysis, or from a court order that forces disclosure of device credentials and stored key material.

Our DH-based constructions address this setting. They remain secure even if all Double Ratchet secrets are disclosed, provided the parties have established a separate *double key* that is not obtained through the same surveillance mechanism.

Partial state disclosure. In some settings, surveillance reveals only part of the ratchet state, while still enabling the adversary to decrypt the contents of normal Signal messages. Full state disclosure requires the authority to capture and maintain the entire Double Ratchet state for every monitored session for direct control over infrastructure or devices.

In practice, many surveillance mechanisms fall short of this. Legal frameworks may authorize access to message contents without mandating disclosure of all cryptographic state, and forensic tools may extract some keys but not others. The cost of recording every ephemeral value for every session may exceed operational budgets. In these cases, the adversary obtains the per-message decryption key and perhaps some additional ratchet state, but not all of it. In particular, the adversary might learn root key material (or information sufficient to derive it) without obtaining the ephemeral DH private keys generated during later ratchet steps. Conversely, the adversary might obtain ephemeral DH private keys from runtime compromise or short-term access, without obtaining the earlier root key material. These patterns also capture operational realities such as incomplete device backups or selective legal authority that grants access to only some session secrets.

Our MAC-based construction exploits this separation. Signal derives the per-message MAC key from secrets that depend on both the root key and DH outputs. If the adversary learns only one side of this state, then the MAC key remains underivable. In that case, covert messages embedded in authentication tags remain confidential and indistinguishable, without requiring any pre-shared double key.

Establishing the double key. The DH-based constructions assume a double key unknown to the adversary. In our threat model, this key is not part of Signal's protocol state and is therefore outside the key material disclosed by the modeled lawful-access mechanism.

For the symmetric variant, both parties must share a double-key secret before covert communication begins. In practice, this can be done through in-person exchange (a passphrase, QR code, or NFC tap), derivation from shared history that predates the surveillance setting, or a one-time bootstrap over a channel outside the adversary's reach (e.g., physical mail or a trusted intermediary).

For the asymmetric variant, only the receiver holds the double secret key, while the corresponding public key can be published. A journalist, activist, or organization can post a double public key on a personal website, enabling any sender to initiate covert communication without prior coordination.

Unless refreshed out of band, we treat the double key as fixed for the corresponding covert context. Thus, the DH-based channel provides transcript undetectability only while the double key remains secret, and does not provide covert-channel forward secrecy or deniability once it is exposed: with the key and transcript, an adversary can recompute and verify past covert contents. A forward-secure variant could derive per-message double keys through a KDF chain analogous to the Double Ratchet, which we leave to future work.

The MAC-based construction requires no double key. It enables covert communication whenever surveillance is partial in the sense above, which makes it applicable even for users who cannot establish an out-of-band secret.

Scope and Non-Goals. Our constructions provide *covert content confidentiality*: the hidden message remains secret, and its presence is computationally undetectable from protocol transcripts under our assumptions.

We do not protect metadata, including who communicates with whom, timing, or frequency. Addressing metadata requires techniques such as anonymous routing or traffic shaping. Our base constructions target surveillance that is passive with respect to message delivery, while our transcript-binding variants address active adversaries who may inject, replay, modify, or reorder protocol messages. We do not address active man-in-the-middle attacks that break endpoint authentication or establish a separate compromised session with either party.

Covert bandwidth. The optimized DH-based channel embeds 32 bits per ratchet step, and the MAC-based channel embeds 64 bits per message. These rates suffice for short sensitive content (identifiers, coordinates, status flags) or for bootstrapping a separate secure channel (e.g., exchanging a session key), but not for bulk transfer.

1.3 Related Work

Secure messaging. End-to-end encrypted messaging aims to keep message contents available only to the communicating endpoints, even when intermediaries relay or store the traffic [28]. Modern secure messaging builds on Off-the-Record Messaging (OTR) [8], which introduced forward secrecy and deniable authentication for interactive conversations. TextSecure [33] brought ratcheted key evolution to mobile messaging and evolved into the Signal protocol, now widely deployed.

Related approaches pursue stronger deniability or concealment goals. Deniable encryption allows coerced users to explain a ciphertext transcript as encrypting alternative plaintexts [10, 39]. Ephemeral messaging systems reduce retained local history through default deletion [50]. Network steganography hides communication within cover traffic or cover distributions [27, 44]. These techniques address important but different threat models: they do not directly study whether covert communication can persist inside the cryptographic values of a deployed E2EE protocol after ordinary message-decryption keys are disclosed.

Signal and the Double Ratchet. Signal combines the X3DH asynchronous key agreement protocol [34] with the Double Ratchet [36]. The Double Ratchet has been analyzed extensively. Alwen et al. [1] gave a modular security treatment, Poettering and Rösler [38] studied compromise resilience, and

Cohn-Gordon et al. [15] proved its guarantees in symbolic and computational models. Bienstock et al. [7] proposed a universally composable framework and introduced the Triple Ratchet for stronger adversarial settings.

Several recent works adapt Signal-style messaging to new threats. Teng and Rasmussen [45] studied in-protocol key verification, while Martiny et al. [35] analyzed Sealed Sender and showed statistical disclosure risks to sender anonymity. PQXDH [31] provides a post-quantum replacement for X3DH. Dodis et al. [20] proposed a Triple Ratchet that combines a post-quantum ratchet with erasure coding, and ML-KEM Braid [40], building on sparse continuous key agreement [3], supports post-quantum continuous key agreement compatible with the Double Ratchet. Dodis et al. [21] studied subversion resistance using Reverse Firewalls to mitigate attacks that tamper with implementations.

Anamorphic encryption. Since the introduction of anamorphic encryption by Persiano et al. [37], subsequent work has refined and extended the foundations. Wang et al. [49] formalized sender-anamorphic encryption for pseudorandom public-key schemes. Banerjee et al. [5] introduced a framework for public-key anamorphic multi-message extensions without relying on randomness recovery. Banfi et al. [6] revisited definitions and proposed stronger security notions. Catalano et al. [12, 13] developed new constructions including homomorphic realizations. Kutyłowski et al. [32] argued that anamorphic channels are inherent to many encryption schemes, framing this as a "self-anti-censorship" property.

A parallel line of work studies *anamorphic-resistant encryption* (ARE), which aims to make covert channels impractical. Dodis and Goldin [19] and Carnemolla et al. [11] proposed ARE schemes with explicit resistance goals. Avitabile et al. [4] gave a practical construction without random oracles and limited covert bandwidth to logarithmically many bits. A more detailed survey appears in the full version [23].

Applications and gap. Anamorphic techniques have been explored beyond messaging, for example in electronic voting [25] and in LLM-generated text channels [26]. Closely related, but not based on anamorphic encryption, Wink [14] provides plausibly deniable messaging over the public channels of existing E2EE applications, including Signal. Wink uses TrustZone to inject encrypted hidden messages into ordinary random coins such as salts and IVs, preserving the application wire format under partial device compromise and compelled disclosure.

Our work is complementary: we instantiate anamorphic encryption directly inside the protocol logic of a deployed E2EE system, namely the Signal Double Ratchet, by embedding covert content in DH ratchet public keys and MAC tags. This gives protocol-native constructions under explicit ratchet-state disclosure models, rather than a deniable-messaging layer outside the ratchet.

1.4 Paper Organization

Section 2 reviews the Double Ratchet and anamorphic encryption. Section 3 formalizes our system model and threat assumptions. Section 4 presents the DH-based constructions, and Section 5 presents the MAC-based construction. Section 6 reports performance measurements. Section 7 discusses limitations and future directions. The full version includes a detailed related-work survey and formal security proofs [23].

2 Background

This work primarily focuses on the two main components of the Double Ratchet protocol, namely, DH and symmetric ratchets. Therefore, we focus on the cryptographic primitives used in these two components, and refer the reader to [36] for additional details. Our description and Figures 1, 2, and 3 follow the Signal Double Ratchet specification [1].

2.1 Cryptographic functions

The Double Ratchet protocol is primarily built upon ECDH key exchange and the HKDF key derivation function [30]. In the following, we assume familiarity with standard cryptographic notions and outline the relevant cryptographic functions, specifying for each its interface, implementation, and key features.

- $\mathsf{Keygen}(\lambda) \to (\mathsf{sk}, \mathsf{pk})$: Given a security parameter λ, the key generation algorithm outputs a key pair $(\mathsf{sk}, \mathsf{pk})$. In the DH ratchet we recall later, it generates an ECDH key pair using either the X25519 or X448 function, denoted as $(\mathsf{sk}, \mathsf{pk} := \mathsf{sk} \cdot G)$, where G is the base point of the elliptic curve, $\mathsf{sk} \in \mathbb{Z}_p$ is a randomly chosen private key, and p is the prime order of the subgroup, denoted as $\mathbb{G}$ with a generator G. We use additive notation for elliptic-curve groups: scalars lie in $\mathbb{Z}_p$, group elements lie in $\mathbb{G}$, and scalar multiplication of a point G by a scalar a is written $a \cdot G$. The default curve in the Signal protocol is Curve25519.
- $\mathsf{DH}(\mathsf{sk}_a, \mathsf{pk}_b) \to \mathsf{dh}_{\mathsf{out}}$: Given A's secret key sk_a and B's public key pk_b, the DH function computes DH shared secret $\mathsf{dh}_{\mathsf{out}} := \mathsf{sk}_a \cdot \mathsf{pk}_b$. Note that $\mathsf{DH}(\mathsf{sk}_a, \mathsf{pk}_b) = \mathsf{DH}(\mathsf{sk}_b, \mathsf{pk}_a)$.
- $\mathsf{PRF}(\mathsf{prf}_{\mathsf{key}}, x) \to y$: Given a key $\mathsf{prf}_{\mathsf{key}}$ and an input x, the pseudorandom function (PRF) outputs an element y. It is required that the output be computationally indistinguishable from a value from a truly random function mapping from the same domain to the same codomain.
- $\mathsf{KDF}(\mathsf{kdf}_{\mathsf{key}}, x) \to y$: Given a key $\mathsf{kdf}_{\mathsf{key}}$ and an input x, the key derivation function outputs an element y. It is required that the output be computationally indistinguishable from a value sampled uniformly at random from the same range. Moreover, the KDF output can

be split into several parts while the knowledge of the partial output should not leak information on the other generated part.

The KDFs for the double ratchet protocol are HMAC-based KDF, consisting of two components: The first extract function takes as input the salt and the input key material, producing a PRF key. Then, the expand function takes the PRF key (along with the implicit auxiliary information and the desired output length) as input, and outputs the derived key. We specify each KDF involved in the ratchet as follows.

- $\mathsf{KDF}_{\mathsf{rk}}(\mathsf{rk}, \mathsf{dh}_{\mathsf{out}}) \rightarrow (\mathsf{rk}_{\mathsf{new}}, \mathsf{ck})$: Given an old root key rk as salt and the shared secret $\mathsf{dh}_{\mathsf{out}}$ as input key material, the HMAC-based key derivation function (HKDF) generates a new root key $\mathsf{rk}_{\mathsf{new}}$ and a new chain key ck.
- $\mathsf{KDF}_{\mathsf{ck}}(\mathsf{ck}) \rightarrow (\mathsf{ck}_{\mathsf{new}}, \mathsf{mk})$: Given the old chain key ck, the HKDF generates a new chain key $\mathsf{ck}_{\mathsf{new}}$ and a message key mk. In this function, a constant value is used as key salt.
- $\mathsf{Enc}(\mathsf{mk}, \mathsf{pt}, \mathsf{ad}) \rightarrow (\mathsf{aes}_{\mathsf{key}}, \mathsf{mac}_{\mathsf{key}}, \mathsf{IV}, \mathsf{ct}, \mathsf{mac})$: Given a message key mk, a plaintext pt and the associated data ad, the encryption algorithm, implemented as Authenticated Encryption with Associated Data (AEAD) in the protocol, outputs an AES key $\mathsf{aes}_{\mathsf{key}}$, a MAC key $\mathsf{mac}_{\mathsf{key}}$, the initialization vector IV, a ciphertext ct and a MAC mac. We include the derived keys in the interface only for analyzing key-disclosure patterns. An ordinary Signal sender outputs only ct and mac. It proceeds in the following three steps, combining CBC and HMAC to ensure misuse resistance in cases where a key is inadvertently reused:
  1. $\mathsf{KDF}_{\mathsf{mk}}(\mathsf{mk}) \rightarrow (\mathsf{aes}_{\mathsf{key}}, \mathsf{mac}_{\mathsf{key}}, \mathsf{IV})$: Given mk as input key material, the HKDF outputs $\mathsf{aes}_{\mathsf{key}}$, $\mathsf{mac}_{\mathsf{key}}$ and IV. The HKDF salt is a zero-filled sequence that matches the output length of the hash function;
  2. $\mathsf{AES}_{\mathsf{CBC}}(\mathsf{aes}_{\mathsf{key}}, \mathsf{IV}, \mathsf{pt}) \rightarrow \mathsf{ct}$: Given $\mathsf{aes}_{\mathsf{key}}$, IV and pt, the encryption function, AES-256 in CBC mode with PKCS#7 padding, outputs a ciphertext ct;
  3. $\mathsf{MAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad}) \rightarrow \mathsf{mac}$: Given $\mathsf{mac}_{\mathsf{key}}$, ct and ad, the MAC function returns a mac.

2.2 The Double Ratchet protocol

Many messaging applications use the Signal's Double Ratchet protocol to generate message keys in support of E2EE. This protocol ensures secure messaging by constantly updating encryption keys. It provides both forward secrecy and post-compromise secrecy. In a nutshell, every time a message is sent, a new key is generated, limiting the exposure of past or future messages if a key is compromised. The protocol bootstraps from a shared secret that sender and receiver have previously agreed based on a secure key agreement protocol. In Signal, this protocol is X3DH [34]. The Double Ratchet protocol consists of the symmetric ratchet and the DH ratchet.

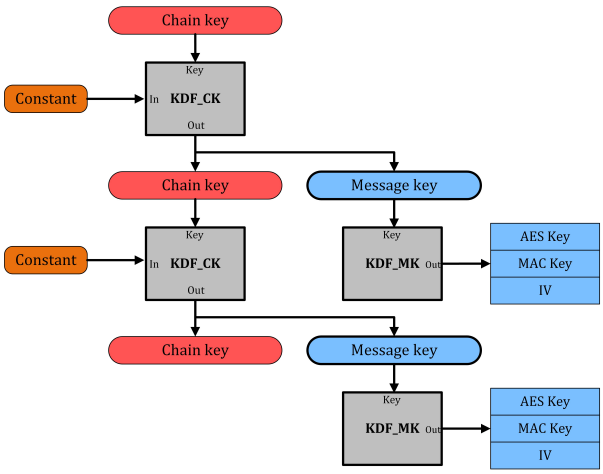


Figure 1: The symmetric ratchet in [1]

Symmetric ratchet. The goal of the symmetric ratchet is to generate a new message key for each message sent or received. It involves the use of the functions $\mathsf{KDF}_{\mathsf{ck}}$ and Enc. The symmetric ratchet initiates with an old chain key, and derives a new chain key and a message key as shown in Figure 1.

DH ratchet. The goal of the DH ratchet is to provide forward secrecy and post-compromise security against an attacker who gains access to a chain key. In the DH ratchet, each party generates a DH key pair, which serves as their current ratchet key pair $(\mathsf{sk}, \mathsf{pk})$. Every message is sent with a header that includes the sender's current public key pk. When one party receives a new public key from the other, they execute DH, replacing their own key pair with a fresh one. This update triggers $\mathsf{KDF}_{\mathsf{ck}}$ to generate a new chain key, ensuring continuous key update for enhanced security. The DH ratchet is illustrated in Figure 2.

In short, the general rules to run either ratchet can be summarised as follows:

- On sending or receiving a new message, a symmetric ratchet step updates the sending or receiving chain to generate a new message key.
- On receiving a new ratchet public key, a DH ratchet step replaces the chain keys before executing the symmetric ratchet.

Combining DH ratchet and symmetric ratchet forms the Double Ratchet protocol as depicted in Figure 3, which illustrates an example of a Double Ratchet scenario from Alice's point of view. In this flow, Alice receives an initial ratchet public key from Bob, then performs a DH ratchet step to obtain her first sending chain key. She then sends two messages within this chain. Subsequently, Bob replies with a new

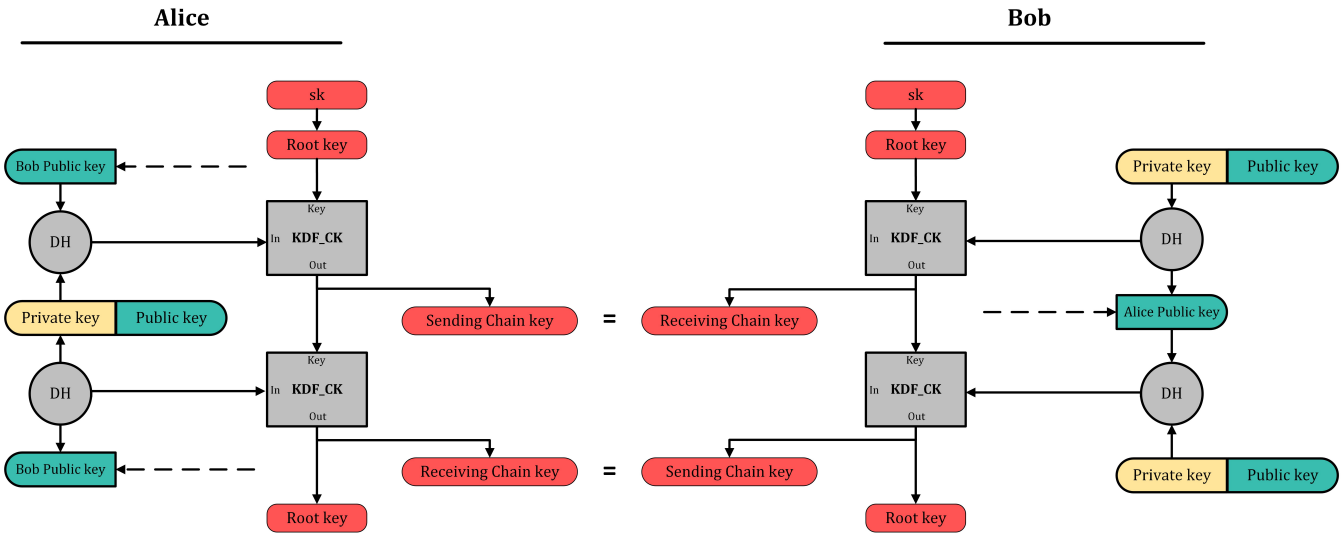


Figure 2: The DH ratchet in [1]

message, including a new ratchet public key. Alice generates a new receiving chain key and derives the corresponding message keys for the messages received from Bob. The final part of this communication involves Alice sending a message along with a new ratchet public key. Note that Bob performs the same operations on his side, generating the corresponding message keys. Since only Alice and Bob share the initial root key generated via X3DH, no one else can compute the sending and receiving keys, and consequently, the message keys. This shared secret also ensures security against man-in-the-middle attacks by the server, which only relays the communication.

Figure 2 also illustrates that compromising a sending or receiving chain key does not affect the security of future keys, as $\mathsf{KDF}_{\mathsf{ck}}$ takes as input the output of $\mathsf{dh}_{\mathsf{out}}$, which is known only to Alice and Bob.

2.3 Anamorphic encryption

Anamorphic encryption [37] protects communication even when a dictator restricts the permitted cryptographic primitives and holds the communication secret key. It augments the standard encryption mode with an anamorphic mode: a sender can use a *double key* to embed a hidden message that only the intended receiver can recover with the same key pair. To the dictator, the security of an anamorphic encryption scheme guarantees that the keys and the ciphertexts in the standard and anamorphic modes are indistinguishable.

An anamorphic encryption scheme linked to a standard encryption scheme of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms $(\mathsf{aGen}, \mathsf{aEnc}, \mathsf{aDec})$. In an anamorphic mode, the aGen algorithm returns an anamorphic public key apk, an anamorphic secret key ask and a double key dk. The aEnc algorithm encrypts a normal message msg together with an anamorphic one amsg with the double key dk into a ciphertext, which can only be decrypted by the aDec algorithm with the double key.

For the Double Ratchet, we design the three anamorphic components in a similar manner. Specifically, there is a double key generation algorithm aGen that samples a double key (pair). Using the double key (pair), the anamorphic message can be encrypted and decrypted. We omit the explicit invocation of double key generation algorithm and assume that, when necessary, the double key pair is generated separately.

3 System Model and Threat Assumptions

This section formalizes the system model, adversary capabilities, and security goals for our anamorphic constructions. We define the key-disclosure patterns under which each construction operates and the security properties we achieve.

3.1 System Model

We consider two parties, Alice and Bob, communicating via the Signal Double Ratchet protocol through a centralized server. The server relays all messages and observes the full protocol transcript, including message headers, ciphertexts, and authentication tags. We assume the underlying cryptographic primitives are instantiated as specified in the Signal documentation [36].

Our constructions augment the standard Double Ratchet with an *anamorphic mode* that embeds covert messages into protocol values. The mechanism for enabling this mode varies by construction and is detailed in subsequent sections.

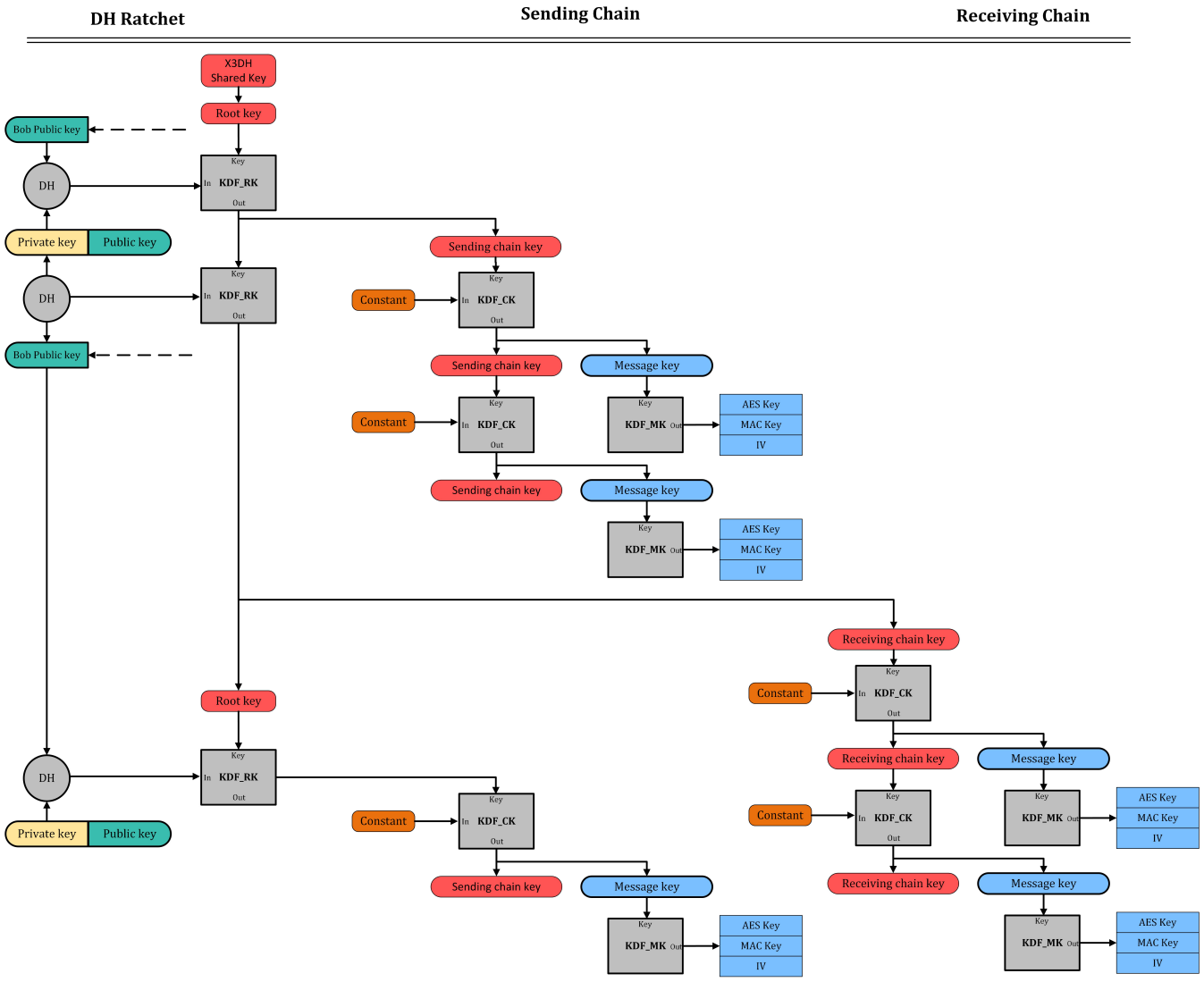


Figure 3: An instance of the Double Ratchet protocol for Alice in [1]

3.2 Adversary Model

We model the adversary $\mathcal{A}$ as controlling the Signal server and obtaining cryptographic state sufficient to decrypt ordinary Signal messages through a key-disclosure mechanism. Formally, $\mathcal{A}$ receives the complete protocol transcript between Alice and Bob, including all message headers and ciphertexts, along with a subset of Double Ratchet state determined by the *key-disclosure pattern* summarized in Table 1.

We consider two adversary classes.

Passive adversary. The adversary observes the transcript and disclosed keys but does not modify, inject, or reorder messages. This models surveillance that decrypts communications without interfering with delivery.

Active adversary. The adversary may additionally inject, replay, modify, or reorder messages. Our base constructions achieve security against passive adversaries. We present transcript-binding upgrades that achieve security against active adversaries.

3.3 Key-Disclosure Patterns

Different surveillance mechanisms expose different subsets of Double Ratchet state. Revealing some keys may enable derivation of others. For instance, exposing both the root key and the DH private keys lets the adversary reconstruct chain and message keys. We identify two principal disclosure patterns corresponding to our two construction families. Throughout, we assume $\mathsf{aes}_{\mathsf{key}}$ is disclosed so that $\mathcal{A}$ can always decrypt ordinary message plaintexts.

Full state disclosure. The adversary learns the complete Double Ratchet state, including the root key, DH private keys,

Table 1: Anamorphic constructions under various threat models. ◊ allows either the root key or the DH private key to be leaked.

Construction	Leak					No PSK required
	Root key	DH private key	Chain/Message key	MAC key	AES key	
Sym-DH	✓	✓	✓	✓	✓	×
Asym-DH	✓	✓	✓	✓	✓	✓
Sym-MAC	◊	◊	×	×	✓	✓

and all derived chain, message, MAC, and AES keys. This pattern arises from comprehensive key-escrow requirements or forensic device compromise.

Our DH-based constructions (Sym-DH and Asym-DH) address this setting. They require a *double key* established outside the Double Ratchet and unknown to $\mathcal{A}$. Covert messages are embedded into DH public keys, which remain indistinguishable from honestly generated keys even when all other state is disclosed. Sym-DH requires pre-sharing the double key between endpoints, while Asym-DH allows the receiver to publish a double public key, enabling covert initiation without prior coordination.

Partial state disclosure. The adversary learns either the root key or the DH private keys, but not both simultaneously. Since chain keys are derived via $\mathsf{KDF}_{\mathsf{rk}}(\mathsf{rk}, \mathsf{DH}(\mathsf{sk}_A, \mathsf{pk}_B))$, partial disclosure prevents the adversary from deriving chain, message, and MAC keys through the standard KDF chain. We assume the adversary still obtains $\mathsf{aes}_{\mathsf{key}}$. This enables decryption of ordinary plaintexts while leaving $\mathsf{mac}_{\mathsf{key}}$ unknown.

This model does not cover full device compromise or disclosure of the whole message key mk. If mk or $\mathsf{mac}_{\mathsf{key}}$ is disclosed, the MAC-based channel is no longer hidden, as captured by the full-state setting. Instead, the model captures surveillance mechanisms whose purpose is content recovery. The authority obtains the value needed to decrypt the visible message, or the resulting plaintext, without receiving the authentication key or all intermediate ratchet secrets. Such a design is sufficient for reading ordinary messages and avoids storing the full evolving Double Ratchet state for every monitored session. It also captures incomplete forensic or backup-based extraction, where one side of the root-key/DH computation is available but the other is not. Under this scoped leakage, $\mathsf{aes}_{\mathsf{key}}$ can be available for ordinary decryption while $\mathsf{mac}_{\mathsf{key}}$ remains underivable.

This pattern captures scenarios such as root key disclosure without access to ephemeral DH private keys (for example, backup-based surveillance that captures long-term state but not runtime ephemeral values), or DH private key disclosure without root key access (for example, short-term device compromise during an active session).

Our MAC-based construction (Sym-MAC) builds on this separation. It requires no pre-shared double key. Covert messages are embedded into authentication tags using the MAC key that $\mathcal{A}$ cannot derive.

Table 1 summarizes the key-disclosure tolerance of each construction. A checkmark indicates that the corresponding key may be disclosed while preserving anamorphic security. The diamond (◊) for Sym-MAC indicates that *either* the root key *or* the DH private key may be leaked, but not both.

3.4 Scope and Assumptions

We consider key-disclosure mechanisms that enable decryption without changing the message format or protocol logic. Surveillance designs that modify other aspects of the Double Ratchet, such as introducing additional protocol messages or altering the ratchet structure, are out of scope because their security impact requires separate analysis.

Our constructions do not change Signal's ordinary out-of-order delivery handling. Covert delivery is tied only to protocol values already carried by delivered Signal messages, namely DH public keys or MAC tags. A reordered or delayed Signal message therefore delays the corresponding covert carrier, but ordinary Signal message processing remains unchanged. For the block-based CCA variants, covert extraction waits until all carriers in the block have been received.

We omit the double key generation process and assume that double keys are established through a separate channel outside of Signal. For Sym-DH, this requires a pre-shared secret. For Asym-DH, the receiver publishes a double public key. For Sym-MAC, no double key is needed.

3.5 Security Goals

For our constructions, we define and prove their *anamorphic security* ensuring that an adversary cannot distinguish the anamorphic mode from the standard mode. That is, the adversary cannot detect whether the observed protocol components contain an anamorphic message or not. We formalize this as follows.

Definition 1 (Anamorphic Security)**.** A construction achieves anamorphic security with respect to a target component T if for every PPT adversary $\mathcal{A}$,

$$\left|\Pr[\mathcal{A}(T^{\mathsf{anam}}, \mathsf{aux}) = 1] - \Pr[\mathcal{A}(T^{\mathsf{std}}, \mathsf{aux}) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where T^{anam} denotes the target component generated in anamorphic mode (embedding a covert message), T^{std} denotes the same component generated in standard mode, and aux captures any additional information available to $\mathcal{A}$ under the relevant key-disclosure pattern. The instantiation of

T and aux is specified per construction. For DH-based constructions, T consists of DH public keys. For the MAC-based construction, T consists of authentication tags.

Anamorphic security in particular yields IND-CPA-style privacy for the covert content. If it is not possible to detect the presence of a covert message, it is also not possible to decide which of two candidate messages was embedded. Moreover, we present transcript-binding upgrades that attain a stronger, CCA-style guarantee against active adversaries. The definition here is an overview. Construction-specific security games and full proofs are provided in the full version [23].

4 Covert Channel in DH Ratchet

To embed covert messages into the DH public keys used in the DH ratchet, we construct an anamorphic channel with a hidden double key. For now, we assume that all keys other than this double key may be leaked. Later in this section, we explain how this assumption is instantiated in different scenarios.

4.1 Base Construction

Recall that in the DH ratchet, fresh DH key pairs are generated regularly. We show how to create *fake* DH key pairs that embed covert messages while remaining indistinguishable from honest ones, and we present two algorithms. The first generates the fake DH key pairs and the second extracts the hidden messages.

Here, we provide an abstract description over an additive prime-order group $\mathbb{G}$ with generator G. A standard DH key generation algorithm samples $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$ and outputs $\mathsf{pk} = \mathsf{sk} \cdot G \in \mathbb{G}$.

Algorithm 1 generates $\ell+1$ key pairs that embed an anamorphic message vector $\overrightarrow{\mathsf{amsg}} \in \mathcal{M}^\ell$ using the double key $\mathsf{dk} \in \{0,1\}^\lambda$. Here, the pseudorandom function PRF with key space $\{0,1\}^\lambda$ maps elements from $\mathbb{G}$ into $\mathbb{Z}_p$.

Algorithm 1 $\mathsf{fakeKeygen}(\mathsf{dk}, \overrightarrow{\mathsf{amsg}})$

parse $\overrightarrow{\mathsf{amsg}} = \{\mathsf{amsg}_i\}_{i=1}^{\ell}$
$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Keygen}(\lambda)$
for $i \in [2, \ell+1]$ **do**
  $\mathsf{sk}_i = \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}) + \mathsf{amsg}_{i-1} \bmod p$
  $\mathsf{pk}_i = \mathsf{sk}_i \cdot G$
end for
return $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i=1}^{\ell+1}$

Algorithm 2 recovers the anamorphic message vector $\overrightarrow{\mathsf{amsg}} \in \mathcal{M}^\ell$ using the double key dk from PK consisting of $\ell+1$ public keys pk_i.

Since recovering anamorphic messages requires solving a discrete logarithm over the admissible message space, $\mathcal{M}$ must be chosen significantly smaller than the group order p. We interpret dlog_G in Algorithm 2 as a bounded search over $\mathcal{M}$: it returns amsg if the residual point equals $\mathsf{amsg} \cdot G$ for some $\mathsf{amsg} \in \mathcal{M}$, and returns $\perp$ otherwise. Under this condition, the construction is robust: decrypting an anamorphic ciphertext under the correct double key yields the embedded message, while an invalid residual point is rejected except with probability at most $|\mathcal{M}|/p$. For Curve25519 and a 32-bit covert payload, this false-accept probability is at most $2^{32}/2^{252} = 2^{-220}$. As we show later, 32-bit payloads are practical with precomputation.

Algorithm 2 $\mathsf{aDec}(\mathsf{dk}, \mathsf{PK})$

parse $\mathsf{PK} = (\mathsf{pk}_i)_{i \in [1,\ell+1]}$
for $i \in [2, \ell+1]$ **do**
  $\mathsf{amsg}_{i-1} = \mathsf{dlog}_G(\mathsf{pk}_i - \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}) \cdot G)$
end for
return $\{\mathsf{amsg}_i\}_{i=1}^{\ell}$

To integrate this into the full Double Ratchet protocol, Alice and Bob perform the following steps to transmit ℓ anamorphic messages within the DH ratchet using $\ell+1$ DH keys. Without loss of generality, we present our construction in the setting where Alice sends covert messages to Bob, and assume for now that Alice and Bob share the same private double key dk.

1. Alice runs a fake key generation algorithm $\mathsf{fakeKeygen}(\mathsf{dk}, \overrightarrow{\mathsf{amsg}})$ as described in Algorithm 1 with her private double key dk.
2. Alice sends $\ell+1$ Signal messages to Bob. Before sending each message, she waits for a successful delivery notification [1].
3. Bob collects the last $\ell+1$ ratchet public keys $\mathsf{PK} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_{\ell+1}\}$ sent by Alice and runs $\mathsf{aDec}(\mathsf{dk}, \mathsf{PK})$ to retrieve $\overrightarrow{\mathsf{amsg}}$ described in Algorithm 2.

We now argue correctness and security of the construction. Intuitively, correctness requires that, given the correct double key dk, the anamorphic messages should be recovered from the anamorphic ciphertexts, that is, from the fake DH public keys produced by our algorithms. Anamorphic security requires that it is infeasible to detect the presence of the covert channels: an adversary should not be able to distinguish honestly generated fresh key pairs and the key pairs that embed anamorphic messages. The formal definitions and proofs are provided in the full version [23].

Proof sketch. Correctness follows because Bob recomputes the same $\mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1})$ value as Alice and subtracts it from pk_i, leaving $\mathsf{amsg}_{i-1} \cdot G$ for discrete-log recovery over the chosen anamorphic message space. For security, replace each $\mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1})$ output by an independent uniform value in

[1] This is necessary to avoid raising suspicion from the server. If Alice sends all $\ell+1$ messages at once, it may appear suspicious that she is not using the same sending chain key for some of them. When Bob sends a Signal message, including a successful delivery notification, he also includes a new DH ratchet public key

$\mathbb{Z}_p$, which changes the adversary's view only by PRF security. Then each secret key $\mathsf{sk}_i = z_i + \mathsf{amsg}_{i-1} \bmod p$ is uniform whenever z_i is uniform, so the corresponding public key $\mathsf{pk}_i = \mathsf{sk}_i \cdot G$ has the honest DH key distribution and hides amsg_{i-1}. The remaining loss comes from possible repeated inputs in the lazy-sampling argument and is bounded by roughly $(\ell Q)^2/p$, which is negligible, over Q oracle queries.

4.2 Stronger Security Guarantee – IND-CCA2

We now strengthen our security guarantee. Analogous to IND-CCA2 security for an encryption scheme, we require that security holds even against an adversary that obtains ciphertexts generated in anamorphic mode and actively interacts with the protocol. In particular, such an adversary must neither be able to distinguish anamorphic executions from honest ones nor learn any information about the embedded anamorphic messages.

This CCA-style variant is block-based. Bob can run the anamorphic decoding and the transcript check only after receiving all DH public keys in the block: the $\ell+1$ keys carrying ℓ covert messages and the additional integrity-check key. Until then, ordinary Signal processing can proceed, but CCA-protected covert extraction is deferred.

Intuitively, the updated fake key generation algorithm binds the covert encoding to the current transcript and adds an extra key pair used as an integrity check. The decryption algorithm is modified accordingly to verify this binding before extracting the anamorphic message. This yields active security at the cost of only one additional key pair.

We update the fake key generation algorithm (Algorithm 1) and the anamorphic message decryption algorithm (Algorithm 2) to Algorithm 3 and Algorithm 4, respectively. We add an auxiliary PRF^* (same key space) that maps the transcript to the secret key, binding keys to the transcript.

Algorithm 3 $\mathsf{fakeKeygen}(\mathsf{dk}, \overrightarrow{\mathsf{amsg}})$

parse $\overrightarrow{\mathsf{amsg}} = \{\mathsf{amsg}_i\}_{i=1}^{\ell}$
$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Keygen}(\lambda)$
for $i \in [2, \ell+1]$ **do**
  $\mathsf{sk}_i = \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}) + \mathsf{amsg}_{i-1} \bmod p$
  $\mathsf{pk}_i = \mathsf{sk}_i \cdot G$
end for
$T = \mathsf{pk}_1 \| \mathsf{pk}_2 \| \ldots \| \mathsf{pk}_{\ell+1}$
$\mathsf{sk}_{\ell+2} = \mathsf{PRF}^*(\mathsf{dk}, T)$
$\mathsf{pk}_{\ell+2} = \mathsf{sk}_{\ell+2} \cdot G$
return $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i=1}^{\ell+2}$

Security claim. Assuming PRF and PRF^* are secure PRFs, the transcript-binding DH variant achieves the CCA-style security goal from Section 3. The first $\ell+1$ public keys are distributed as in the base construction, so they remain indistinguishable from honest DH public keys. The extra public key

Algorithm 4 $\mathsf{aDec}(\mathsf{dk}, \mathsf{PK})$

parse $\mathsf{PK} = (\mathsf{pk}_i)_{i \in [1,\ell+2]}$
$T = \mathsf{pk}_1 \| \mathsf{pk}_2 \| \ldots \| \mathsf{pk}_{\ell+1}$
if $\mathsf{pk}_{\ell+2} \neq \mathsf{PRF}^*(\mathsf{dk}, T) \cdot G$ **then**
  return $\perp$
end if
for $i \in [2, \ell+1]$ **do**
  $\mathsf{amsg}_{i-1} = \mathsf{dlog}_G(\mathsf{pk}_i - \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}) \cdot G)$
end for
return $\overrightarrow{\mathsf{amsg}}$

binds the block to $T = \mathsf{pk}_1 \| \cdots \| \mathsf{pk}_{\ell+1}$ through $\mathsf{PRF}^*(\mathsf{dk}, T)$. Therefore, any fresh block that an active adversary modifies, injects, or replays is accepted only if its final public key equals $\mathsf{PRF}^*(\mathsf{dk}, T) \cdot G$, where T is the transcript derived from the received block. Without dk, producing such a value for a new transcript is a PRF forgery, which succeeds only with negligible probability. The formal game-based definition and proof are given in the full version [23].

4.3 Instantiation of Double Key dk

The DH-based construction assumes Alice and Bob hold a double key dk unknown to the adversary. We now describe two instantiations corresponding to different key-establishment settings.

Symmetric variant (Sym-DH). Both parties share a pre-shared secret dk established out-of-band. Alice uses dk to generate fake DH keys encoding covert messages, and Bob uses dk to recover them. The adversary, who observes the full Double Ratchet state but not dk, cannot detect or decode the covert channel.

Asymmetric variant (Asym-DH). The receiver Bob holds a double key pair $(\mathsf{dsk}_B, \mathsf{dpk}_B)$ and publishes the public component dpk_B. In the same sense, Alice also publishes her public key dpk_A. Alice embeds covert messages using dpk_B, and Bob recovers them using dsk_B, as they both can compute a common "double key" used in the algorithms via $\mathsf{dk} = \mathsf{DH}(\mathsf{dsk}_B, \mathsf{dpk}_A) = \mathsf{DH}(\mathsf{dsk}_A, \mathsf{dpk}_B)$. This removes the need for prior coordination and suits settings where publishing a long-term public key is feasible (e.g., a journalist or organization posting a double public key on their website).

4.4 Optimization

As discussed before, naively recovering anamorphic messages requires an exhaustive search over the anamorphic message space, so $\mathcal{M}$ must be small. We now present optimized variants of Algorithms 1 and 2 that use a precomputed lookup table, which is also used in our implementation. Intuitively, the optimized algorithms optimize the anamorphic decryption by replacing an exhaustive search with a single group operation and a table lookup. We successfully tested the feasibility of this optimisation for values up to 32 bits. However, it

can be extended further, since the main computational effort lies in precomputing the lookup table only once. The same lookup-table technique extends to the construction with a stronger security guarantee in a similar way. The following is a concrete instantiation over Curve25519.

Table 2: Notation used in Algorithms 5 and 6

Symbol	Description
G_E	Base point of the Edwards curve E.
P_E and Q_E	Points on the Edwards curve E.
$\mathsf{f}_{\mathrm{E}\to\mathrm{M}}$	Map from Edwards curve to Montgomery representation. Note that this function provides the extra bit of information sign needed for inversion, since the Montgomery model does not retain sign information.
$\mathsf{f}_{\mathrm{M}\to\mathrm{E}}$	Map from Montgomery to Edwards representation, using the supplied choice of sign for the Edwards point.
$\mathsf{lsb}(\mathsf{pk}_i)$	Least significant bit of pk_i.
$\mathsf{lookup}(\mathsf{Q}_E)$	Lookup table mapping an Edwards curve point to a message.

In Algorithm 5, each fake secret key is chosen as a PRF-derived scalar (keyed by dk and the previous public key) plus the anamorphic message. The corresponding public key is then selected so that its least significant bit determines the correct sign to invert back to the unique Edwards point. This ensures that the public keys are distributed as in the honest scheme, while encoding a small message-dependent offset in a reversible way.

Algorithm 5 $\mathsf{fakeKeygen}(\mathsf{dk}, \overrightarrow{\mathsf{amsg}}, \ell)$

$(\mathsf{sk}_1, \mathsf{pk}_1) = \mathsf{Keygen}(\lambda)$
for $i \in [2, \ell+1]$ **do**
  $\mathsf{ctr} = 0$
  repeat
    $\mathsf{sk}_i = \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}, \mathsf{ctr}) + \mathsf{amsg}_{i-1} \bmod p$
    $\mathsf{P}_E = \mathsf{sk}_i \cdot G_E$
    $(\mathsf{pk}_i, \mathsf{sign}) = \mathsf{f}_{\mathrm{E}\to\mathrm{M}}(\mathsf{P}_E)$
    $\mathsf{ctr} = \mathsf{ctr} + 1$
  until $\mathsf{lsb}(\mathsf{pk}_i) = \mathsf{sign}$
end for
return $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i=1}^{\ell+1}$

Algorithm 6 uses dk to recompute and subtract the PRF-derived scalar from the recovered Edwards point, yielding a point of the form $\mathsf{amsg} \cdot G_E$. Since we precomputed a lookup table mapping all candidate points $\mathsf{amsg} \cdot G_E$ to their corresponding messages, the decryption of each anamorphic block requires only one scalar multiplication and a table lookup, without iterating over all possible messages. This makes, for example, a 32-bit anamorphic message space efficiently practical in our implementation. The correctness and security guarantees of the original construction remain unchanged.

Algorithm 6 $\mathsf{aDec}(\mathsf{dk}, \mathsf{PK})$

$\mathsf{PK} = (\mathsf{pk}_i)_{i\in[1,\ell+1]}$
for $i \in [2, \ell+1]$ **do**
  $\mathsf{ctr} = 0$
  repeat
    $x = \mathsf{PRF}(\mathsf{dk}, \mathsf{pk}_{i-1}, \mathsf{ctr})$
    $\mathsf{P}_E = \mathsf{f}_{\mathrm{M}\to\mathrm{E}}(\mathsf{pk}_i, \mathsf{lsb}(\mathsf{pk}_i))$
    $\mathsf{Q}_E = \mathsf{P}_E - x \cdot G_E$
    $\mathsf{amsg}_{i-1} = \mathsf{lookup}(\mathsf{Q}_E)$
    $\mathsf{ctr} = \mathsf{ctr} + 1$
  until $\mathsf{amsg}_{i-1} \neq \perp$
end for
return $\{\mathsf{amsg}_i\}_{i=1}^{\ell}$

Precomputation technique. The lookup table is precomputed by calculating $\mathsf{Q}_E = \mathsf{amsg} \cdot G_E$ for each amsg in the anamorphic message space and storing the mapping $\mathsf{Q}_E \mapsto \mathsf{amsg}$. This speeds up repeated point-to-index queries by performing the expensive elliptic-curve computations in advance. The deterministic lookup table is built over a fixed range of candidate values (amsg), storing for each value the corresponding curve point in compressed form Q_E together with its index. The table is then sorted by the compressed point encoding, so that a query only requires compressing the target point and performing a binary search to recover the matching index. This replaces repeated full-range curve computations with a one-time preprocessing step and fast file-based lookups, making the method both efficient and easy to reproduce across systems. The robustness bound above follows from the lookup table covering only the admissible message space rather than the full group.

5 Covert Channel in Symmetric Ratchet

To embed hidden messages into the symmetric ratchet, we create an anamorphic channel based on the MAC key that remains unknown to the server. Only the intended receiver, who holds this key, can recover the hidden messages as part of the integrity verification. This construction, namely *Sym-MAC*, targets scenarios where *either* the root key *or* the private DH ratchet key is leaked to the server, but not both, so that the resulting chain, message, and MAC keys are never fully exposed. From the adversary's view, the MACs carrying hidden messages are indistinguishable from ordinary MACs. No pre-shared key or other public double key is required, as the anamorphic messages are embedded using MAC keys derived within the Double Ratchet.

5.1 Base Construction

We first describe two algorithms that specify how to generate *fake* MACs and how to extract hidden messages from them.

Algorithm 7 embeds an anamorphic message $\mathsf{amsg} \in \mathcal{M}$ into a MAC tag under key $\mathsf{mac}_{\mathsf{key}}$ by XORing amsg with an honestly computed MAC. Algorithm 8 recovers the anamorphic message $\mathsf{amsg} \in \mathcal{M}$ by recomputing the honest MAC tag under $\mathsf{mac}_{\mathsf{key}}$ and extracting amsg via XOR.

Algorithm 7 $\mathsf{fakeMAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad}, \mathsf{amsg})$

$t = \mathsf{MAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad})$
$\mathsf{mac} = t \oplus \mathsf{amsg}$
return mac

Algorithm 8 $\mathsf{aDec}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad}, \mathsf{mac})$

$t = \mathsf{MAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad})$
if $\mathsf{mac} = t$ **then**
    skip ▷ Not anamorphic
else
    $\mathsf{amsg} = t \oplus \mathsf{mac}$
    return amsg
end if

In this construction, we eliminate the small message-space constraint required by the covert channel construction in DH keys. For the base construction, the anamorphic message space is $\mathcal{M} = \{0,1\}^{\lambda} \setminus \{0^{\lambda}\}$, where 0^{λ} is reserved to denote no covert payload because it yields the honest MAC tag. In our implementation, $\lambda = 64$, matching the MAC size specified in the Signal Double Ratchet documentation. This restriction applies only to the base construction. The active variant below can encode an all-zero payload block because its final binding tag identifies the block as anamorphic.

MAC verification and rejection. A receiver that runs the unmodified Signal verification procedure would reject an anamorphic tag, because $\mathsf{mac} \neq \mathsf{MAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad})$ whenever $\mathsf{amsg} \neq 0$. In our construction, the intended receiver runs aDec with the same $\mathsf{mac}_{\mathsf{key}}$ before treating the mismatch as an ordinary authentication failure: equality means the tag is ordinary, while a mismatch is interpreted as an anamorphic payload. This behavior is invisible to the adversary in the partial-disclosure model, because $\mathsf{mac}_{\mathsf{key}}$ is not disclosed. If $\mathsf{mac}_{\mathsf{key}}$ is disclosed, the adversary can recompute the honest tag and detect the modification, so Sym-MAC does not apply. In the active variant below, the final tag authenticates the block and causes modified non-anamorphic tags to be rejected.

To integrate this into the full Double Ratchet protocol, Alice and Bob perform the following steps to transmit one anamorphic message amsg along with one normal Signal message m.

1. Alice encrypts a normal Signal message m into $\mathsf{ct} = \mathsf{AES}_{\mathsf{CBC}}(\mathsf{aes}_{\mathsf{key}}, \mathsf{IV}, m)$ and computes a tag as described in Algorithm 7 instead of the original mac.
2. Bob detects and recovers the anamorphic message amsg by running aDec as shown in Algorithm 8.

We now argue for the correctness and security of the construction. Correctness requires that, given the MAC key $\mathsf{mac}_{\mathsf{key}}$, any valid hidden message can be recovered; this follows directly from the invertibility of the XOR operation.

Anamorphic security requires that MACs carrying hidden messages be indistinguishable from honestly generated MACs. This holds as long as $\mathsf{mac}_{\mathsf{key}}$ remains secret and the underlying MAC produces pseudorandom tags, since XOR with a fixed message preserves their indistinguishability from random. As illustrated in Figure 1, secrecy of $\mathsf{mac}_{\mathsf{key}}$ implies secrecy of the corresponding message key and, in turn, of the chain key. Because deriving the chain key requires both the DH output and the root key, the construction remains secure even if either the private ratchet keys or the root key is compromised, as long as they are not both leaked simultaneously. The formal definition and proof are provided in the full version [23].

Proof sketch. Correctness follows directly from XOR: Bob recomputes the honest tag $t = \mathsf{MAC}(\mathsf{mac}_{\mathsf{key}}, \mathsf{ct}, \mathsf{ad})$ and obtains $\mathsf{amsg} = \mathsf{mac} \oplus t$. For security, under an undisclosed $\mathsf{mac}_{\mathsf{key}}$, the honest tag t is pseudorandom to the adversary. The anamorphic tag $t \oplus \mathsf{amsg}$ is therefore also pseudorandom for any fixed covert message amsg, so it is indistinguishable from an honest MAC tag. In the Double Ratchet instantiation, the partial-disclosure condition preserves this argument because the adversary cannot derive $\mathsf{mac}_{\mathsf{key}}$ unless it obtains both the root key and the DH secret needed by the KDF chain.

5.2 Stronger Security Guarantee – IND-CCA2

Similar to our previous construction, here, we also require that security holds even against an adversary that obtains ciphertexts generated in anamorphic mode and actively interacts with the protocol.

This CCA-style variant is block-based. Bob can run the anamorphic decoding and the transcript check only after receiving all $\ell + 1$ MAC tags in the block. Until then, ordinary Signal message processing can proceed, but CCA-protected covert extraction is deferred.

Intuitively, Alice embeds ℓ anamorphic messages, each of length λ, within $\ell + 1$ normal Signal messages. The $\ell + 1$ MAC tags are generated so that the first ℓ tags are used in the symmetric ratchet to carry the corresponding covert payloads, and the final tag authenticates the preceding transcript to provide an integrity check.

We update the fake MAC algorithm (Algorithm 7) and the anamorphic message decryption algorithm (Algorithm 8) to Algorithm 9 and Algorithm 10, respectively. We add an

auxiliary PRF (same key space as MAC key) that maps the transcript to the tag.

Algorithm 9 $\mathsf{fakeMAC}(\overrightarrow{\mathsf{mac_{key}}}, \overrightarrow{\mathsf{ct}}, \overrightarrow{\mathsf{ad}}, \overrightarrow{\mathsf{amsg}})$

for $i \in [1,\ell]$ **do**
  $\mathsf{mac}_i = \mathsf{MAC}(\mathsf{mac}_{\mathsf{key}_i}, \mathsf{ct}_i, \mathsf{ad}_i) \oplus \mathsf{amsg}_i$
end for
$T = \mathsf{mac}_1 \| \ldots \| \mathsf{mac}_\ell$
$\mathsf{mac}_{\ell+1} = \mathsf{PRF}(\mathsf{mac}_{\mathsf{key}_{\ell+1}}, T)$
return $\overrightarrow{\mathsf{mac}}$

Algorithm 10 $\mathsf{aDec}(\overrightarrow{\mathsf{mac_{key}}}, \overrightarrow{\mathsf{ct}}, \overrightarrow{\mathsf{ad}}, \overrightarrow{\mathsf{mac}})$

$\overrightarrow{t} = \mathsf{MAC}(\overrightarrow{\mathsf{mac_{key}}}, \overrightarrow{\mathsf{ct}}, \overrightarrow{\mathsf{ad}})$
if $\mathsf{mac}_i = t_i$ for $i \in [1,\ell+1]$ **then**
  skip ▷ Not anamorphic
else
  for $i \in [1,\ell]$ **do**
    $\mathsf{amsg}_i = \mathsf{mac}_i \oplus t_i$
  end for
  $T = \mathsf{mac}_1 \| \ldots \| \mathsf{mac}_\ell$
  $\sigma = \mathsf{PRF}(\mathsf{mac}_{\mathsf{key}_{\ell+1}}, T)$
  if $\sigma = \mathsf{mac}_{\ell+1}$ **then**
    return $\overrightarrow{\mathsf{amsg}}$
  else
    return $\perp$
  end if
end if

Alice and Bob perform the following steps to transmit ℓ anamorphic messages $\overrightarrow{\mathsf{amsg}}$ within $\ell+1$ normal Signal messages $\overrightarrow{m}$.

1. Alice encrypts $\ell+1$ normal Signal messages $\overrightarrow{m}$ into $\overrightarrow{\mathsf{ct}} = \mathsf{AES_{CBC}}(\overrightarrow{\mathsf{aes_{key}}}, \overrightarrow{\mathsf{IV}}, \overrightarrow{m})$ where $\overrightarrow{\mathsf{aes_{key}}}$ and $\overrightarrow{\mathsf{IV}}$ are generated honestly along the KDF chain.
2. Alice then computes the tags as described in Algorithm 9 and uses them as MACs corresponding to the messages instead of the original $\overrightarrow{\mathsf{mac}}$.
3. Bob detects and recovers the anamorphic message amsg by running aDec as shown in Algorithm 10.

Security claim. Assuming the MAC and PRF outputs are pseudorandom under undisclosed $\mathsf{mac_{key}}$, the transcript-binding MAC variant achieves the CCA-style security goal from Section 3. The first ℓ tags remain indistinguishable from honest MAC tags by the base construction. The final tag is computed as $\mathsf{PRF}(\mathsf{mac}_{\mathsf{key}_{\ell+1}}, T)$ on the received transcript value. Thus, an active adversary who modifies, injects, or replays a block must produce the correct final tag for the resulting transcript before the block is accepted. Since $\mathsf{mac}_{\mathsf{key}_{\ell+1}}$ is hidden under the partial-disclosure condition, this reduces to forging a fresh PRF output and succeeds only with negligible probability. The formal game-based definition and proof are given in the full version [23].

6 Performance

We implement prototypes of our anamorphic messaging constructions [24] in the official Signal application [43] using Rust, Java, and Kotlin. We empirically evaluate their impact across different covert bit-rate settings, both in the official Signal-Android client [42] and in libSignal [41], the platform-agnostic library used by official Signal clients and mainly written in Rust.

We run our experiments on an Android device running Android 14 with a Qualcomm Snapdragon 888 processor and 8 GB of RAM, and on Android Studio emulators hosted on a 2023 MacBook Pro with an M2 Pro processor and 16 GB of RAM.

For the optimized DH-based construction, we compute the lookup table using an Intel Xeon Gold 6248R 3 GHz with 96 cores and 800 GB of RAM. The 16-bit lookup table is generated in a few seconds and takes 2.3 MB of space, while the 32-bit lookup table is generated in around 50 minutes and takes 144 GB of space. Both time and space complexity are exponential. However, the lookup table needs to be computed only once and can be exposed through a remote query service if local storage is an issue. Our optimized implementation supports remote lookup-table queries, allowing clients to avoid local table storage.

Table 3 presents performance results for experiments conducted on a real Android device and an Android emulator. We report the average encryption and decryption times for random messages using each anamorphic messaging construction, alongside the original Signal protocol.

We also evaluate the worst-case decryption time, which occurs when a message does *not* contain an anamorphic payload. In addition, we measure the round-trip time on the recipient side, defined as the duration between receiving the message from the server and sending the corresponding delivery receipt back. This metric is important for assessing potential timing side channels, as the time taken to generate and send the receipt could reveal to the server whether the message was anamorphic.[2]

As decryption is the most computationally intensive part of our constructions and a key performance indicator, we evaluate decryption time in both Signal-Android and libSignal. To evaluate aDec in our constructions (Sym-DH, Asym-DH,

[2] One could avoid sending the "delivery" message at all. In such a case, the DH ratchet would not be updated, since the delivery message carries a fresh public key, and the sender would need to wait for a message reply from the receiver before sending a new anamorphic message in the Sym-DH and Asym-DH constructions. In addition, not sending a delivery message might appear suspicious to the server, as sending a delivery message is standard in Signal.

Table 3: Performance in milliseconds on a real Android device (top) and Android emulator (bottom).

Construction	Anamorphic length	Signal-Android				libSignal		
		Decryption		Round-trip		Encryption	Decryption	
		Avg	Worst	Avg	Worst	Avg	Avg	Worst
Android real device								
Original	None	1.56	—	20.5	—	1.53	1.42	—
Base (A)Sym-DH	16-bit	822.8	1622.2	841.48	1673.28	1.71	811.34	1617.41
Sym-MAC	64-bit	1.57	—	20.6	—	1.53	1.40	—
Android emulator								
Original	None	1.50	—	120.10	—	1.11	1.11	—
Base (A)Sym-DH	16-bit	438.2	888.2	567.3	1010.2	1.81	440.3	883.2
Optimized (A)Sym-DH	16-bit	7.01	14.12	149.96	150.14	1.13	5.77	13.20
	32-bit	7.22	65.10	150.32	200.18	1.32	5.43	65.88
Sym-MAC	64-bit	1.51	—	120.15	—	1.11	1.12	—

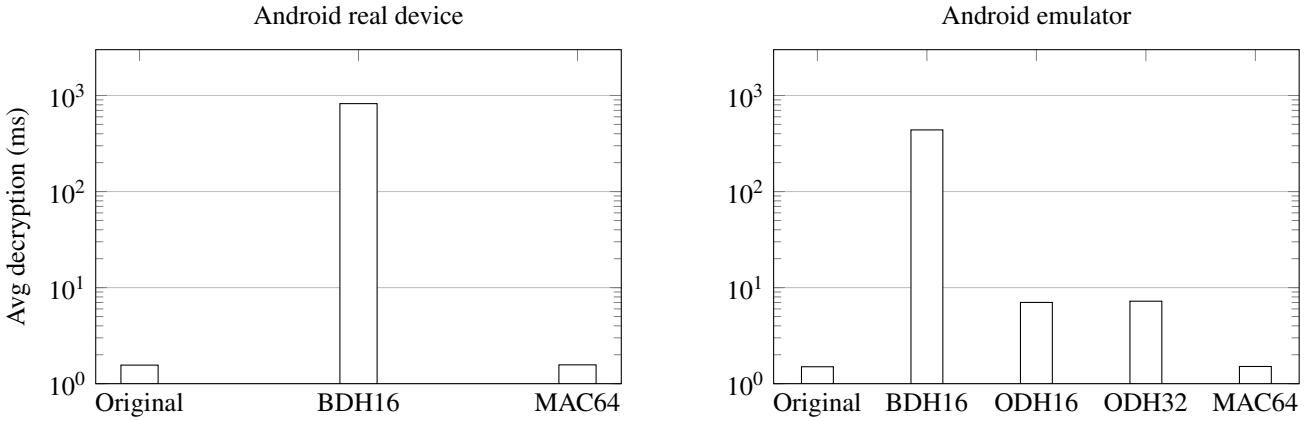


Figure 4: Average decryption latency measured in the Signal-Android library on the Android emulator and real Android device, grouped by construction and anamorphic bit-length (log scale). BDH16, MAC64, ODH16, and ODH32 stand for Base (A)Sym-DH 16-bit, Sym-MAC 64-bit, Optimized (A)Sym-DH 16-bit, and Optimized (A)Sym-DH 32-bit, respectively.

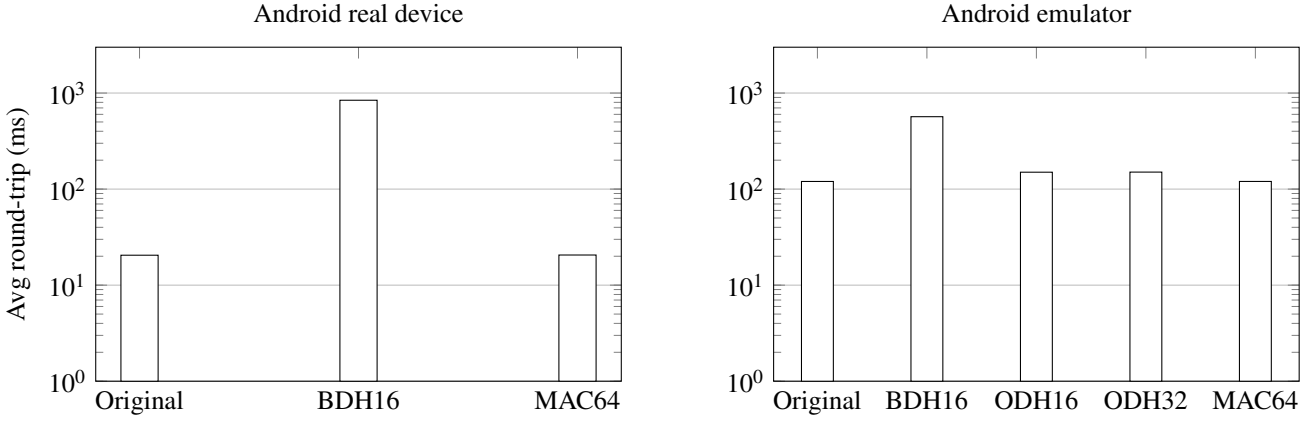


Figure 5: Average round-trip latency measured in the Signal-Android library on the Android emulator and real Android device, grouped by construction and anamorphic bit-length (log scale).

and Sym-MAC), we implemented the base variants. We also implemented the optimized (A)Sym-DH variants on the Android emulator to show their performance advantage. Note that the PRFs in Sym-DH and Asym-DH are instantiated using HMAC-SHA256.

In general, message encryption times are not affected by the generation of anamorphic messages. Looking at libsignal on the real device, the original Signal protocol takes approximately 1.53 ms to encrypt a message, while all anamorphic constructions require nearly the same time to encrypt an anamorphic message, regardless of its size. Similar results can be observed on the Android emulator, as shown in Table 3.

Decryption time depends on the construction. This is expected, as decryption may involve iterative operations, particularly the discrete-logarithm computation required to recover the anamorphic message in the Sym-DH and Asym-DH constructions. However, there is no significant average difference within the Optimized (A)Sym-DH construction, nor between decryption in the core libSignal library and in the Signal-Android client. In particular, the optimized (A)Sym-DH construction avoids exhaustive on-the-fly discrete-logarithm search and has only 7x decryption overhead. The user-visible round-trip time in the optimized version stays close to baseline Signal.

The round-trip time, as measured in the Signal-Android code, varies considerably only in the base (A)Sym-DH construction, while it stays close to the original Signal implementation in both the Sym-MAC and optimized (A)Sym-DH constructions. Therefore, the Sym-MAC construction is the most efficient according to all indicators, but it supports the weakest threat model. The optimized (A)Sym-DH construction supports the strongest threat model and performs within the same order of magnitude as the original Signal construction. The base (A)Sym-DH construction takes significantly longer than the original Signal protocol, as the receiver must search over up to 16 bits of anamorphic information using multiple scalar multiplications, as shown in Figures 4 and 5. This construction is therefore suitable only when a lookup table cannot be maintained. In this case, to mitigate side-channel risks, the receiver can send the delivery receipt after a fixed delay matching the average round-trip time in the original implementation, rather than waiting for successful decryption. However, the sender should transmit each message using an interval longer than the worst-case round-trip time, approximately 1600 ms on the real device and 1000 ms on the emulator.

7 Conclusions

In this work, we presented two constructions for anamorphic channels for the Signal Double Ratchet protocol: a DH-based channel that embeds payloads in DH key pairs and a MAC-based channel that embeds payloads in authentication tags of the symmetric ratchet. Our designs remain compatible with existing Signal-style deployments, require no server changes, and operate even when one or more Double Ratchet keys are disclosed, for example as a result of a surveillance mechanism. Our analysis and implementations show that the DH-based and MAC-based channels can efficiently carry at least 32 and 64 hidden bits per Signal message, respectively.

To our knowledge, this is the first practical instantiation of anamorphic encryption within a deployed E2EE protocol. The Signal Double Ratchet underpins communication for billions of users, many of whom, including journalists, activists, lawyers, and people in vulnerable situations, depend on its confidentiality guarantees in contexts where the consequences of exposure are severe.

Several authorities have pushed Signal and other E2EE messaging platforms to enable surveillance by breaking E2EE confidentiality to snoop on messages. Our results indicate that even if decryption keys are exposed, covert channels that preserve normal protocol behavior can still exist.

Forward secrecy for the covert channel is outside the scope of our constructions. If the DH-channel double key is later exposed, an adversary with the transcript can recompute and verify prior covert contents. A natural direction is to derive per-message double keys through a KDF chain analogous to the Double Ratchet. Anamorphic authenticated key exchange [48] provides one possible starting point.

A key direction for future work concerns the potential deployment of Anamorphic-Resistant Encryption schemes. If ARE systems become efficient and widely used, it will be important to understand whether some form of anamorphism can still be maintained, and under what leakage or functionality constraints. In particular, studying ARE-style defenses in the context of the Signal protocol would clarify the minimal conditions under which covert channels remain possible.

Finally, our approach is worth investigating in protocols that reuse Double Ratchet–style building blocks, including recent post-quantum proposals such as Triple Ratchet [20] and Apple's PQ3 [2]. Beyond messaging, similar embeddings may apply to other E2EE services, including file sharing and real-time audio and video, offering a clearer path to supporting E2EE even under key-disclosure pressure.

Ethical Considerations

Our work sits at the intersection of privacy technology and regulatory debate. We embed covert communication channels into the Signal Double Ratchet protocol, enabling users to exchange hidden messages even when an adversary can decrypt their regular communications. We present a stakeholder-based ethics analysis following the USENIX Security guidelines.

Stakeholders

Our work primarily concerns people who need private communication under surveillance. This includes journalists pro-

tecting sources, activists, and ordinary citizens in countries where encryption may be compromised by law. For these users, our constructions offer a layer of protection when traditional E2EE fails.

Developers and maintainers of messaging platforms gain a clearer understanding of where covert channels can persist inside widely deployed protocols, which informs threat modeling and security audits. However, platforms may also face pressure to deploy countermeasures, which could introduce new security issues through misinformed mitigations.

For policymakers and law enforcement, our work provides technical input to ongoing debates. We demonstrate that decryptability does not imply detectability, and that covert channels survive even under comprehensive key disclosure. This matters for realistic assessment of what surveillance mandates can achieve.

Finally, we recognize the dual-use concern directly. The same techniques protecting vulnerable users could conceal harmful communications by malicious actors. We address this below.

Human Subjects, Data, and Platform Interaction

This research involves no human subjects, no user studies, and no collection of private data. We do not access or monitor real user communications. All experiments run in controlled settings on our own Android devices and emulators, measuring computational overhead and timing metrics.

We do not probe or interact with live Signal infrastructure. Our implementation modifies client and library code only in a research context. We did not contact Signal for vulnerability disclosure because we found no vulnerability. Our constructions extend the protocol without weakening existing guarantees.

Risks and Potential Harms

The primary ethical concern is dual-use. Covert channels can be exploited by malicious actors to evade legitimate oversight. However, anamorphic encryption already exists in the cryptographic literature. Our contribution is applying these techniques to a deployed protocol, not inventing new capabilities.

More broadly, publishing practical techniques may provoke stronger surveillance countermeasures with negative societal consequences.

Mitigations

We are careful about what we claim. Our formal security proofs show indistinguishability under stated assumptions. Our experiments measure timing and overhead. We also specify which keys must remain secret for each construction. Users must assess which threat model fits their situation.

To reduce the risk of enabling misuse, our goal is to inform security research and policy rather than provide deployment-ready tools. We release code to support reproducibility of our measurements and figures, and we avoid packaging a drop-in covert messaging feature or operational guidance aimed at evading oversight for end-user deployment.

Rationale for Conducting the Research

We conducted this research because key-disclosure and lawful-access proposals often assume that decrypting overt messages is sufficient to eliminate private communication. Testing this assumption requires analyzing the protocol components where covert channels could persist. We limited the work to controlled experiments on our own devices and emulators, without interacting with live Signal infrastructure or collecting user data. We considered the dual-use risk before implementation and scoped the artifact toward reproducibility of the paper's claims rather than an end-user covert messaging tool.

Rationale for Publication

Publication is justified because it addresses a question with direct policy relevance: under key-disclosure models, does decrypting overt messages eliminate covert communication? Our answer is no. Covert channels persist in protocol elements like DH keys and MAC tags. This finding benefits implementers, auditors, and policymakers by enabling informed decisions.

We acknowledge residual dual-use risk. We believe the benefits to vulnerable populations, scientific understanding, and policy clarity outweigh this risk. The barrier to actual deployment is high, requiring pre-shared secrets, modified clients on both ends, and active cooperation between parties.

Open Science

Our code artifact is available on Zenodo at https://doi.org/10.5281/zenodo.20425141.

Acknowledgements

Research at the University of Luxembourg was supported by the project Security and Verifiability of Electronic Testing and Assessment Systems (SEVERITAS, INTER_ANR_20_14926102_SEVERITAS).

This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

References

[1] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: security notions, proofs, and modularization for the signal protocol. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–158. Springer, 2019.

[2] Apple Security Engineering and Architecture. iMessage with PQ3: The new state of the art in quantum-secure messaging at scale. https://security.apple.com/blog/imessage-pq3/, 2024. Apple Security Research, 21 February 2024. Accessed: 2026-06-01.

[3] Benedikt Auerbach, Yevgeniy Dodis, Daniel Jost, Shuichi Katsumata, and Rolfe Schmidt. How to compare bandwidth constrained two-party secure messaging protocols: A quest for a more efficient and secure post-quantum protocol. In *Proceedings of the 34th USENIX Security Symposium*, 2025.

[4] Gennaro Avitabile, Vincenzo Botta, Emanuele Giunta, Marcin Mielniczuk, and Francesco Migliaro. The malice of elfs: practical anamorphic-resistant encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 303–333. Springer, 2026.

[5] Shalini Banerjee, Tapas Pal, Andy Rupp, and Daniel Slamanig. Simple Asymmetric Anamorphic Encryption and Signature Using Multi-Message Extensions. In *Annual International Cryptology Conference*. Springer, 2026. To appear. Available as Cryptology ePrint Archive, Paper 2025/370.

[6] Fabio Banfi, Konstantin Gegier, Martin Hirt, Ueli Maurer, and Guilherme Rito. Anamorphic encryption, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–32. Springer, 2024.

[7] Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. A more complete analysis of the signal double ratchet algorithm. In *Annual International Cryptology Conference*, pages 784–813. Springer, 2022.

[8] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, pages 77–84, 2004.

[9] Ran Canetti, Palak Jain, Marika Swanberg, and Mayank Varia. Universally composable end-to-end secure messaging. In *Annual International Cryptology Conference*, pages 3–33. Springer, 2022.

[10] Rein Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Annual International Cryptology Conference*, pages 90–104. Springer, 1997.

[11] Davide Carnemolla, Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic resistant encryption: the good, the bad and the ugly. In *Annual International Cryptology Conference*, pages 472–503. Springer, 2025.

[12] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic encryption: New constructions and homomorphic realizations. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–62. Springer, 2024.

[13] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Generic anamorphic encryption, revisited: New limitations and constructions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 275–303. Springer, 2025.

[14] Anrin Chakraborti, Darius Suciu, and Radu Sion. Wink: deniable secure messaging. In *32nd USENIX security symposium (USENIX Security 23)*, pages 1271–1288, 2023.

[15] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, 2020.

[16] Daniel Collins, Doreen Riepel, and Si An Oliver Tran. On the tight security of the double ratchet. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4747–4761, 2024.

[17] Daniel Croft. Signal threatens to withdraw from Australia as government pushes for backdoor data access. https://www.cyberdaily.au/security/12437-signal-threatens-to-withdraw-from-australia-as-government-pushes-for-backdoor-data-access, 2025. Cyber Daily, 30 July 2025. Accessed: 2026-06-01.

[18] Digital Watch Observatory. Sweden considers law requiring encrypted messaging backdoors, Signal threatens to exit. https://dig.watch/updates/sweden-considers-law-requiring-encrypted-messaging-backdoors-signal-threatens-to-exit, 2025. 26 February 2025. Accessed: 2026-06-01.

[19] Yevgeniy Dodis and Eli Goldin. Anamorphic-resistant encryption; or why the encryption debate is still alive. In *Annual International Cryptology Conference*, pages 440–471. Springer, 2025.

[20] Yevgeniy Dodis, Daniel Jost, Shuichi Katsumata, Thomas Prest, and Rolfe Schmidt. Triple ratchet: A bandwidth efficient hybrid-secure signal protocol. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 302–331. Springer, 2025.

[21] Yevgeniy Dodis, Bernardo Magri, Noah Stephens-Davidowitz, and Yiannis Tselekounis. Guarding the signal: secure messaging with reverse firewalls. In *Annual International Cryptology Conference*, pages 39–74. Springer, 2025.

[22] European Commission. ProtectEU: A European Internal Security Strategy. COM(2025) 148 final, CELEX:52025DC0148, 2025. Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions, 1 April 2025. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52025DC0148.

[23] Rosario Giustolisi, Gabriele Lenzini, Chuanwei Lin, Mohammadamin Rakeei, and Andy Rupp. Crypto wars in secure messaging: Covert channels in signal despite leaked keys. Cryptology ePrint Archive, Paper 2025/2172, 2025. URL: https://eprint.iacr.org/2025/2172.

[24] Rosario Giustolisi, Gabriele Lenzini, Chuanwei Lin, Mohammadamin Rakeei, and Andy Rupp. Crypto Wars in Secure Messaging: Covert Channels in Signal Despite Leaked Keys. Zenodo, https://doi.org/10.5281/zenodo.20425141, May 2026. Software artifact.

[25] Rosario Giustolisi, Mohammadamin Rakeei, and Gabriele Lenzini. Anamorphic voting: Ballot freedom against dishonest authorities. In *International Conference on Financial Cryptography and Data Security. FC 2025 International Workshops*, pages 266–280. Springer, 2025.

[26] Danilo Gligoroski and Mayank Raikwar. An llm framework for cryptography over chat channels. In *Proceedings of the Workshop on Privacy in Large Language Models (LLM) and Natural Language Processing (NLP) 2025*, pages 32–43, 2025.

[27] Gabriel Kaptchuk, Tushar M Jois, Matthew Green, and Aviel D Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1529–1548, 2021.

[28] Mallory Knodel, Sofía Celi, Olaf Kolkman, and Gurshabad Grover. Definition of end-to-end encryption. Cryptology ePrint Archive, Paper 2024/2085, 2024. URL: https://eprint.iacr.org/2024/2085.

[29] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European symposium on security and privacy (EuroS&P)*, pages 435–450. IEEE, 2017.

[30] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *Annual Cryptology Conference*, pages 631–648. Springer, 2010.

[31] Ehren Kret and Rolfe Schmidt. The PQXDH Key Agreement Protocol. https://signal.org/docs/specifications/pqxdh/, 2024. Revision 3, 2023-05-24, last updated 2024-01-23.

[32] Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. The self-anti-censorship nature of encryption: On the prevalence of anamorphic cryptography. *Proceedings on Privacy Enhancing Technologies*, 2023(4):170–183, 2023.

[33] Moxie Marlinspike. Advanced cryptographic ratcheting. https://signal.org/blog/advanced-ratcheting/, 2013. Signal Blog, 26 November 2013. Accessed: 2026-06-01.

[34] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. https://signal.org/docs/specifications/x3dh/, 2016. Revision 1, 2016-11-04.

[35] Ian Martiny, Gabriel Kaptchuk, Adam J Aviv, Daniel S Roche, and Eric Wustrow. Improving signal's sealed sender. In *NDSS*, 2021.

[36] Trevor Perrin, Moxie Marlinspike, and Rolfe Schmidt. The Double Ratchet Algorithm. https://signal.org/docs/specifications/doubleratchet/, 2025. Revision 4, 2025-11-04.

[37] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: private communication against a dictator. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 34–63. Springer, 2022.

[38] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2018.

[39] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In

Proceedings of the forty-sixth annual ACM symposium on Theory of computing, pages 475–484, 2014.

[40] Rolfe Schmidt. The ML-KEM Braid Protocol. https://signal.org/docs/specifications/mlkembraid/, 2025. Revision 1, 2025-02-21, last updated 2025-09-26.

[41] Signal. libsignal. https://github.com/signalapp/libsignal, 2025. Github repository, Accessed: May 2025.

[42] Signal. Signal-android. https://github.com/signalapp/Signal-Android, 2025. Github repository, Accessed: May 2025.

[43] Signal. signalapp. https://github.com/signalapp, 2025. Github repository, Accessed: May 2025.

[44] Gustavus J Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of Crypto 83*, pages 51–67. Springer, 1984.

[45] Wil Liam Teng and Kasper Rasmussen. Defending against attack on the cloned: In-band active man-in-the-middle detection for the signal protocol. *arXiv preprint arXiv:2410.16098*, 2024.

[46] Chris Vallance. Signal would "walk" from UK if Online Safety Bill undermined encryption. https://www.bbc.com/news/technology-64584001, 2023. BBC News, 24 February 2023. Accessed: 2026-06-01.

[47] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the signal protocol. In *International Conference on Applied Cryptography and Network Security*, pages 188–209. Springer, 2020.

[48] Weihao Wang, Shuai Han, and Shengli Liu. Anamorphic authenticated key exchange: Double key distribution under surveillance. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 168–200. Springer, 2024.

[49] Yi Wang, Rongmao Chen, Xinyi Huang, and Moti Yung. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 135–167. Springer, 2023.

[50] Bin Xu, Pamara Chang, Christopher L Welker, Natalya N Bazarova, and Dan Cosley. Automatic archiving versus default deletion: what snapchat tells us about ephemerality in design. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*, pages 1662–1675, 2016.