

# Tinkering with Cryptography

<http://blu.org/cgi-bin/calendar/2023-mar>

Meeting Notes: <http://blu.org/meetings/2023/03/>  
(There were great comments from the chat!)

Added  
After  
Meeting



Boston Linux & UNIX was originally founded in 1994 as part of The Boston Computer Society. We meet on the third Wednesday of each month at the Massachusetts Institute of Technology, in Building E51.

<a href="#">Home</a>	<a href="#">Calendar</a>	<a href="#">Mail Lists</a>	<a href="#">List Archives</a>	<a href="#">Desktop SIG</a>	<a href="#">Photos</a>	<a href="#">Video</a>	<a href="#">Maps/Directions</a>	<a href="#">InstallFests</a>	<a href="#">PGP Key signings</a>
<a href="#">Linux Cafe/BBQ</a>	<a href="#">Notes</a>	<a href="#">Articles</a>	<a href="#">Blog</a>	<a href="#">IRC Channel</a>	<a href="#">Links</a>	<a href="#">BLU Apparel</a>	<a href="#">About BLU</a>		

## Tinkering With Cryptography

Date and Time

Wednesday, March 15, 2023 from 6:30 pm to 9:00 pm

Location

[Online at Jitsi](#)

Presenters



[Brian DeLacey](#) - *CryptographicHistory@gmail.com*

Summary

Making cryptography more accessible and useful with Google's Tink Cryptographic Library

Abstract

Brian covers the basics of a relatively new library from Google, with demonstration code running on multiple platforms. We'll also delve into some of the more advanced topics related to this code and take a special, deep dive into Key Management Systems.

As part of the discussion, Brian explores the challenges of brittle bytes and how to achieve secure, authenticated access to critical data over time. Demos cover code running on tiny little machines and bigger&mdash;but still bargain&mdash;builds.

Demonstration code will be in Golang and Python, but the Tink Cryptographic Library also works well with C++, Java, mobile platforms and more.

We'll also walk through and demonstrate code and the use of cryptography in "nostr", which stands for "Notes and Other Stuff Transmitted by Relays".

According to its chief architect, nostr is "The simplest open protocol that is able to create a censorship-resistant global "social" network once and for all. It doesn't rely on any trusted central server, hence it is resilient; it is based on cryptographic keys and signatures, so it is tamperproof; it does not rely on P2P techniques, therefore it works."

<http://blu.org/cgi-bin/calendar/2023-mar>

<https://developers.google.com/tink>

<https://github.com/aljazceru/awesome-nostr>

<https://nostr.com/>

<https://github.com/nostr-protocol/nostr>

<https://www.apple.com/newsroom/2022/12/apple-advances-user-security-with-powerful-new-data-protections/>



# DISCLAIMER

*This discussion was NOT intended to be a security review of any specific cryptographic libraries. Rather, it was intended to be a tinkering view of working with some of the available tools. For that, you may find other available papers to be relevant.*

## Security of Streaming Encryption in Google's Tink Library

Viet Tung Hoang<sup>1</sup> and Yaobin Shen<sup>2</sup>

<sup>1</sup> Dept. of Computer Science, Florida State University

<sup>2</sup> Dept. of Computer Science & Engineering, Shanghai Jiao Tong University, China

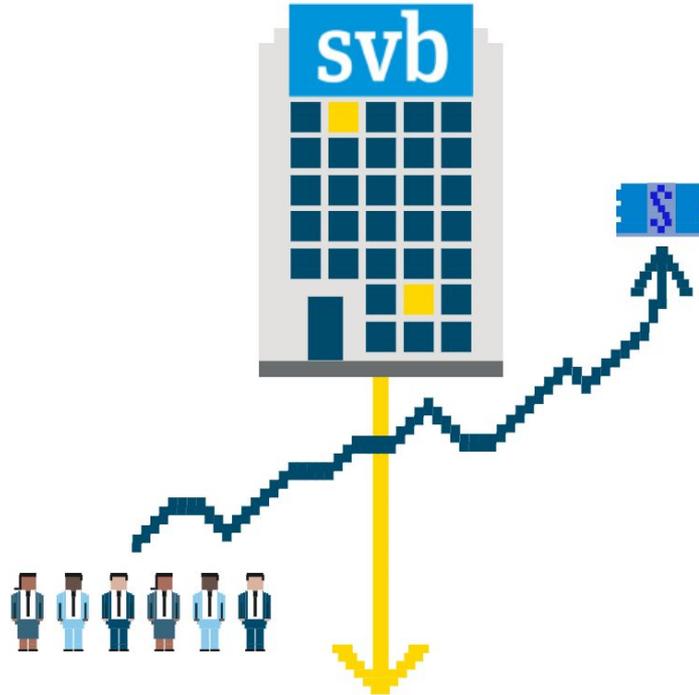
August 23, 2020

<https://eprint.iacr.org/2020/1019.pdf>

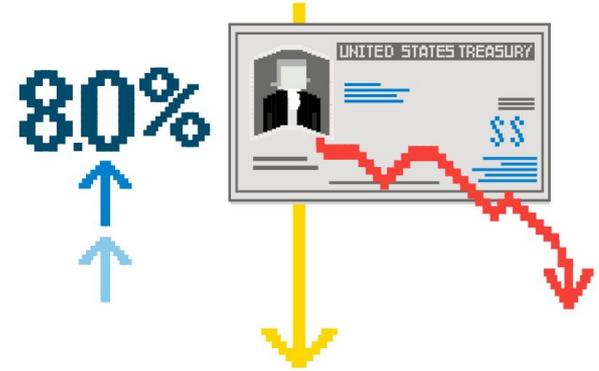
Added  
After  
Meeting



# SVB

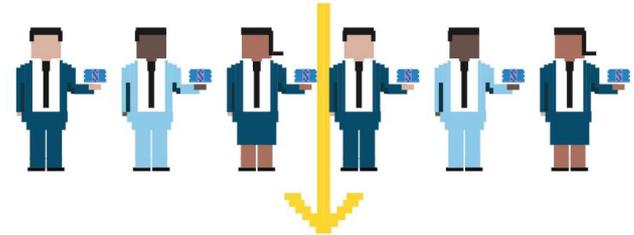


<https://www.usatoday.com/story/graphics/2023/03/13/graphics-bank-collapse-silicon-valley/11466073002/>

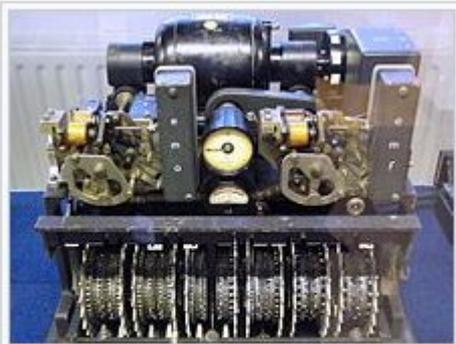


The market reacted sharply and SVB lost over \$160 billion dollars in value in 24 hours.

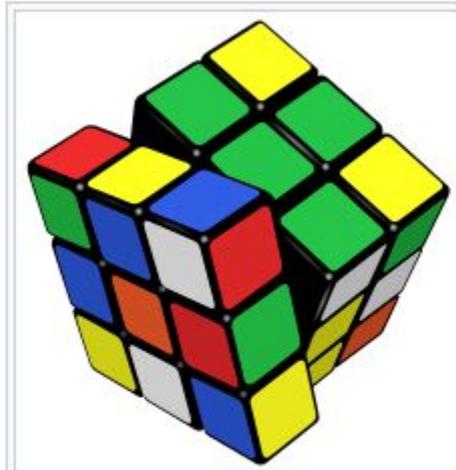
As the stock fell, depositors moved quickly to withdraw money from the bank.



# Are Rubik, Crypto & Group Theory Related?



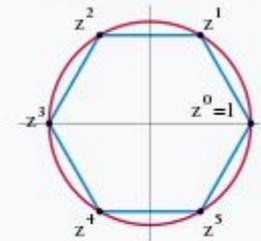
Lorenz cipher machine, used in World War II to encrypt communications of the German High Command



The manipulations of the Rubik's Cube form the Rubik's Cube group.

Algebraic structure – Group theory

## Group theory



Basic notions [\[show\]](#)

Finite groups [\[show\]](#)

Discrete groups · Lattices [\[show\]](#)

Topological and Lie groups [\[show\]](#)

Algebraic groups [\[show\]](#)

V · T · E

# Tink Crypto Library (Goog)



## What is Tink?

Tink is an open-source cryptography library written by cryptographers and security engineers at Google. Tink's secure and simple APIs reduce common pitfalls through user-centered design, careful implementation and code reviews, and extensive testing.

Tink helps users without a cryptography background safely implement common cryptographic tasks. At Google, Tink has been deployed in hundreds of products and systems.

<https://developers.google.com/tink>

# Why should I use Tink?

## It's easy to use

Cryptography is difficult to get right. With Tink, you can [encrypt](#) or [sign data](#) with just a few lines of code, with built-in security guarantees to help you avoid pitfalls.

## It's secure

Tink adds security protections on top of well known libraries like BoringSSL and Java Cryptography Architecture and shows them right in the interfaces, so auditors and tools can quickly find gaps. Tink also separates APIs that are potentially dangerous, so you can monitor them.

## It's compatible

Tink ciphertexts are compatible with existing cryptography libraries. Tink also supports [encrypting or storing keys](#) in Amazon KMS, Google Cloud KMS, Android Keystore, and iOS Keychain.

▼ I want to...

Encrypt data

Encrypt large files or data streams

Exchange data

Protect data from tampering

Sign data

Use client-side encryption with a cloud provider

**Key management**

Overview

▶ Protect keys with an external KMS

▶ Use Tinkey to manage keys

Generate an encrypted keyset

Generate a plaintext keyset

▶ Key management best practices

**Advanced topics**

I want to encrypt data deterministically

I want to protect structured data

I want to bind ciphertext to its context

I want to meet FIPS 140-2 requirements

I want to learn about the Tink wire format



# Tink Design

Goals

[https://developers.google.com/tink/design/goals\\_of\\_tink](https://developers.google.com/tink/design/goals_of_tink)

Primitives and Interfaces

[https://developers.google.com/tink/design/primitives\\_and\\_interfaces](https://developers.google.com/tink/design/primitives_and_interfaces)

Keys

<https://developers.google.com/tink/design/keys>

Keysets

<https://developers.google.com/tink/design/keysets>



# How to Use Tink

I want to encrypt data

<https://developers.google.com/tink/encrypt-data>

[https://github.com/google/tink/blob/master/go/aead/aead\\_test.go](https://github.com/google/tink/blob/master/go/aead/aead_test.go)

# I want to encrypt data

For most users and use cases, the Authenticated Encryption with Associated Data (AEAD) primitive is the simplest and most appropriate to implement. AEAD offers guarantees of secrecy and authenticity, and ensures that messages always have different ciphertexts (encrypted outputs) even if the plaintext messages (the inputs for the encryption) are the same. It uses a single key for both encryption and decryption.

**We recommend using the AES128\_GCM key type for most data encryption use cases.** For all supported key types, see [Supported Key Types](#).

The following examples get you started using the AEAD primitive.

★ **Note:** If you want to see examples that use Go or Objective-C, see our how-to documentation in [GitHub](#).

```
Python Java C++ Go
python/examples/aead/aead_basic.py View on GitHub

import tink
from tink import aead
from tink import cleartext_keyset_handle

def example():
    """Encrypt and decrypt using AEAD."""
    # Register the AEAD key managers. This is needed to create an Aead primitive
    # later.
    aead.register()

    # A keyset created with "tinkey create-keyset --key-template=AEAD256_GCM". Note
    # that this keyset has the secret key information in cleartext.
    keyset = r"""
key: [{
  "keyData": {
    "keyMaterialType":
      "SYMMETRIC",
    "typeUrl":
      "type.googleapis.com/google.crypto.tink.AesGcmKey",
    "value":
      "G18BYUfGgYk3RTRhj/LIUzSudIw1yjCftCOypTr8jONSLg=="
  },
  "keyId": 294406504,
  "outputPrefixType": "TINK",
  "status": "ENABLED"
}],
"primaryKeyId": 294406504
}"""

    # Create a keyset handle from the cleartext keyset in the previous
    # step. The keyset handle provides abstract access to the underlying keyset to
    # limit access of the raw key material. WARNING: In practice, it is unlikely
    # you will want to use a cleartext_keyset_handle, as it implies that your key
    # material is passed in cleartext, which is a security risk.
    keyset_handle = cleartext_keyset_handle.read(tink.JsonKeysetReader(keyset))

    # Retrieve the Aead primitive we want to use from the keyset handle.
    primitive = keyset_handle.primitive(aead.Aead)

    # Use the primitive to encrypt a message. In this case the primary key of the
    # keyset will be used (which is also the only key in this example).
    ciphertext = primitive.encrypt(b'msg', b'associated_data')

    # Use the primitive to decrypt the message. Decrypt finds the correct key in
    # the keyset and decrypts the ciphertext. If no key is found or decryption
    # fails, it raises an error.
    output = primitive.decrypt(ciphertext, b'associated_data')
```

```
Python Java C++ Go
go/aead/aead_test.go View on GitHub

import (
    "bytes"
    "fmt"
    "log"

    "github.com/google/tink/go/aead"
    "github.com/google/tink/go/insecurecleartextkeyset"
    "github.com/google/tink/go/keyset"
)

func Example() {
    // A keyset created with "tinkey create-keyset --key-template=AEAD256_GCM". Note
    // that this keyset has the secret key information in cleartext.
    jsonKeyset := `{
        "key": [{
            "keyData": {
                "keyMaterialType":
                    "SYMMETRIC",
                "typeUrl":
                    "type.googleapis.com/google.crypto.tink.AesGcmKey",
                "value":
                    "G18BYUfGgYk3RTRhj/LIUzSudIw1yjCftCOypTr8jONSLg=="
            },
            "keyId": 294406504,
            "outputPrefixType": "TINK",
            "status": "ENABLED"
        }],
        "primaryKeyId": 294406504
    }`

    // Create a keyset handle from the cleartext keyset in the previous
    // step. The keyset handle provides abstract access to the underlying keyset to
    // limit the exposure of accessing the raw key material. WARNING: In practice,
    // it is unlikely you will want to use an insecurecleartextkeyset, as it implies
    // that your key material is passed in cleartext, which is a security risk.
    // Consider encrypting it with a remote key in Cloud KMS, AWS KMS or HashiCorp Vault.
    // See https://github.com/google/tink/blob/master/docs/GOLANG-HOWTO.md#storing-and-loading
    keysetHandle, err := insecurecleartextkeyset.Read(
        keyset.NewJSONReader(bytes.NewReader(string(jsonKeyset)))
    )
    if err != nil {
        log.Fatal(err)
    }

    // Retrieve the AEAD primitive we want to use from the keyset handle.
    primitive, err := aead.New(keysetHandle)
    if err != nil {
        log.Fatal(err)
    }

    // Use the primitive to encrypt a message. In this case the primary key of the
    // keyset will be used (which is also the only key in this example).
    plaintext := []byte("message")
    associatedData := []byte("associated data")
    ciphertext, err := primitive.Encrypt(plaintext, associatedData)
    if err != nil {
        log.Fatal(err)
    }

    // Use the primitive to decrypt the message. Decrypt finds the correct key in
    // the keyset and decrypts the ciphertext. If no key is found or decryption
    // fails, it returns an error.
    decrypted, err := primitive.Decrypt(ciphertext, associatedData)
    if err != nil {
        log.Fatal(err)
    }

    fmt.Println(string(decrypted))
    // Output: message
}
```

# Tink

---

## **NOTE: Tink is moving!**

As part of our roadmap we are splitting Tink into [multiple GitHub repositories](#) that will be hosted at [github.com/tink-crypto](https://github.com/tink-crypto) and will be independently versioned.

Roughly, we are going to create one repository per language, library extension such as KMS (except Tink Python), and tools.

A few important highlights:

- The migration will be done gradually over the course of 2023 with a new release from each of the new repositories. Releases will be announced in our [mailing list](#).
- We will keep updating each implementation/tool in [github.com/google/tink](https://github.com/google/tink) for a specified amount of time; migrated implementations/tools will eventually stop being updated on [github.com/google/tink](https://github.com/google/tink). The support window depends on the specific implementation, as shown in the table below.
- New issues and pull requests should be created in the new repos.

Tink implementation/extension	New repository	Migration status	End of support in google/tink
Tink Java	<a href="#">tink-crypto/tink-java</a>	In progress (Q1 2023)	Q3 2023
Tink Java AWS KMS extension	<a href="#">tink-crypto/tink-java-awskms</a>	In progress (Q1 2023)	Q3 2023
Tink Java Google Cloud KMS extension	<a href="#">tink-crypto/tink-java-gcpkms</a>	In progress (Q1 2023)	Q3 2023
Tink Java apps extension	<a href="#">tink-crypto/tink-</a>	In progress (Q1	Q3 2023

Tink C++	Tink Python	<a href="#">tink-crypto/tink-py</a>	Not started (expected Q2 2023)	TBA
Tink C++ AWS KMS extension				
Tink C++ Google Cloud KMS extension	Tink Go	<a href="#">tink-crypto/tink-go</a>	Not started (expected Q3 2023)	TBA
Tink Python				
Tink Go	Tink Go AWS KMS extension	<a href="#">tink-crypto/tink-go-awskms</a>	Not started (expected Q3 2023)	TBA
Tink Go AWS KMS extension				
Tink Go Google Cloud KMS extension	Tink Go Google Cloud KMS extension	<a href="#">tink-crypto/tink-go-gcpkms</a>	Not started (expected Q3 2023)	TBA
Tink Go HashiCorp Vault KM extension				
Tink Javascript				
Tink Obj-C				
Tink Tinkey				
Tink cross language tests	<a href="#">tink-crypto/tink-cross-lang-tests</a>	Not started (expected Q4 2023)	TBA	



# Tink Link

## Cryptographic Function

A *cryptographic function* is a map

$$f : \mathbf{K} \times \mathbf{R} \times \mathbf{I} \rightarrow \mathbf{O}$$

from a set  $\mathbf{K}$  (the key space), a set  $\mathbf{R} = \{0, 1\}^\infty$  (randomness, which we assume to be the set of infinite bitstrings), and a set  $\mathbf{I}$  (the input space), to a set  $\mathbf{O}$  (the output space).

Goals

[https://developers.google.com/tink/design/goals\\_of\\_tink](https://developers.google.com/tink/design/goals_of_tink)

Primitives and Interfaces

[https://developers.google.com/tink/design/primitives\\_and\\_interfaces](https://developers.google.com/tink/design/primitives_and_interfaces)

fiatjaf Retweeted



**Kollider** ⚡ @kollider\_trade · 2h



Today we're excited to announce the launch of Kollider Relay, the first subscription-based paid relay on Nostr.

Quick thread on what it is and how to use it.



Welcome to **Kollider**

# Nostr Relay

Subscription

## 8000 Sats/Month

- High Performance
- Low Latency
- Subscription Based

[Learn more](#)

[Fill Pubkey from Wallet](#)

[Get Invoice](#)

[Need Help? Contact Us](#)



**fiatjaf** @fiatjaf · Mar 14

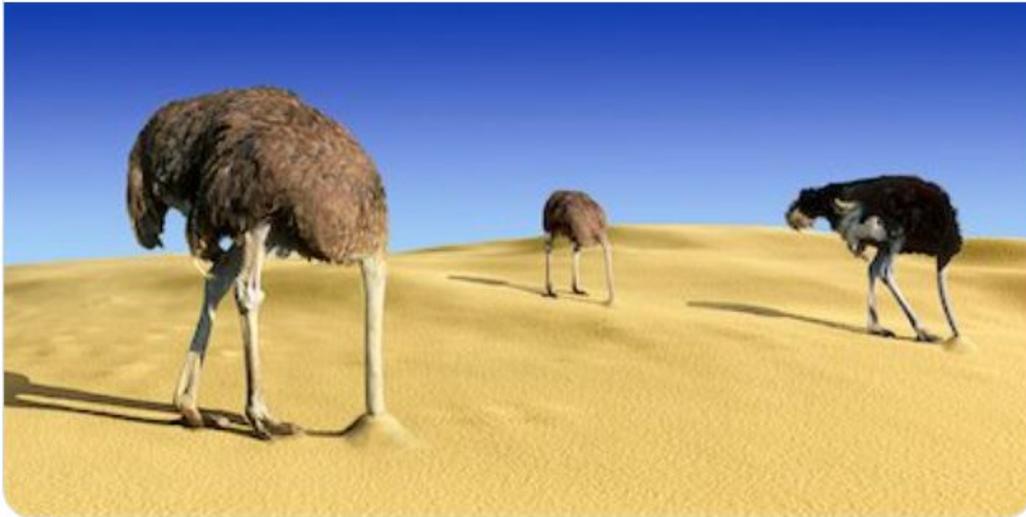


A monstrous thread complaining about Nostr I hadn't seen. Good read!

**Level39** @level39 · Mar 4

📖 1) Nostr is 👍, but there are limitations and serious risks associated with using Nostr for your identity. Let's take a closer look at what the issues are, why it needs to be solved and why using Nostr for identity can increase the risk of security vulnerabilities and attacks.

[Show this thread](#)





**Level39** @level39 · Mar 4



Replying to [@level39](#)

2) The ability to robustly secure one's identity is critical to functioning societies. Without proof of identity, individuals cannot access banking services, gain employment, or receive voting rights. Solving decentralized identity is crucial for individual sovereignty.





**Level39** @level39 · Mar 4



3) There are 1.1 billion people who are not able to formally prove their identity — the majority of which live in Africa and Asia. For any decentralized social media protocol to be fully inclusive, worldwide, it must be able to robustly support identity for the unidentified.





Level39 @level39 · Mar 4



4) Currently, most online federated identities are controlled by centralized and permissioned entities, such as Facebook. Google, Mozilla, Apple and Twitter. Users can easily find themselves deplatformed by these companies, for a variety of reasons.

## Sign In



Continue with Facebook

ERROR



Continue with Google



Continue with Apple

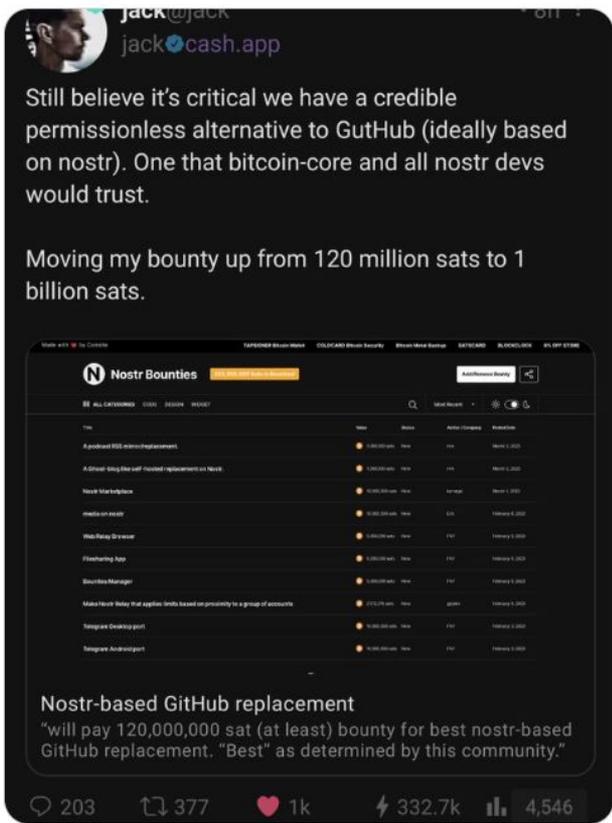
or continue with email



Level39 @level39 · Mar 4



5) @jack has a bounty for a trusted Nostr-based GitHub. Nostr keys are being considered as a way to manage online identities. However, as we will see, using Nostr keys for identity can introduce significant security vulnerabilities and risks for users. [bountsr.org/code/2023/01/1...](https://bountsr.org/code/2023/01/1...)



Level39 @level39 · Mar 4



6) The Nostr protocol provides users with a public and a private key. The user's identity is tied to this public key, as opposed to a username. Much like your Bitcoin, if you lose or leak your private key your Nostr identity is gone.





Level39 @level39 · Mar 4



7) The average Bitcoin holder might be able to recover from a loss of some funds. However, losing one's identity can be far more catastrophic — particularly if that identity is used for one's career, unlocks access to other applications or signing powers.





**Level39** @level39 · Mar 4



8) Recently, [@BTCGandalf](#) accidentally shared his private Nostr key on Twitter, instead of his public key. It was an easy mistake that anyone could have made and is indicative of UX issues with the protocol. This gave everyone control of his Nostr account and his Nostr identity.



**GANDALF**  @BTCGandalf · Feb 14



Accidentally copied my NOSTR private key instead of my public key and share it on here.

Was using a third party app to post to Twitter so didn't realise until I was warned via other channels.

RIP my NOSTR account - people have already accessed it and posted from it.

 103

 27

 348

 82.9K



39

Level39 @level39 · Mar 4

...

9) To solve the issue, @BTCGandalf had to go back to Twitter, disavow his Nostr identity, and announce a new Nostr key as his new identity. He plans to tie his identity to DNS records. However, if his DNS were to expire or be seized, his identity would be permanently captured.



GANDALF @BTCGandalf · 20h

...

ok ok last one and then I'm out again for the week.

Come hang on NOSTR and follow along as I struggle to verify my account using my own domain 🤔

npub1ac8qr6chl3ktnfdjvqd97y5tdgs2eg579tvd0rfdydhgjtzcqnr657s



**LOSING YOUR  
NOSTR PRIVATE  
KEY IN A  
"BOATING ACCIDENT"**



**ACCIDENTALLY  
TWEETING YOUR  
NOSTR PRIVATE KEY**

imgflip.com

39

Level39 @level39 · Mar 4

...

10) Nostr is a communications protocol, not an identity protocol. As such, it does not solve identity at the protocol level. Thus, it is in Nostr users' best interest to have maximum interoperability with existing standardized decentralized identity protocols.





**Level39** @level39 · Mar 4



11) Nostr handles identity "out of band" — meaning that your identity within the protocol entirely relies on external identities. Nostr simply points to centralized or seizable sources of identity and everyone presumes that those sources are trustworthy and accurate.





Level39 @level39 · Mar 4

12) For example, when @jack first joined Nostr, he put his public key on [Cash.app](#), which allowed anyone to see it was him. However, this approach is permissioned. If a user doesn't own the domain hosting their identifier, it can be easily deplatformed.

@jb55

This is how you know it's the real jack. [cash.app](#) points to his pubkey, verified by @damusapp. Anyone can do this!

12:57 50S

jack  
@jack [cash.app](#)

npub1sg6p1zptd64:f2371mu63q0uf63m

bitcoin...twtr/@jack

111 Following Followers 9 Relays

jack @jack 1min  
Changing my NIP5

jack

jack@cash.app  
winningblowgun86@walletofsatoshi.com  
npub1sg6p1zptd64:f2371mu63q0uf63m  
#bitcoin

NIP05 identifier <https://cash.app/well-known/nostr.json?name=jack>

```
"names": {  
  "jack": "82341f882b6eabcd2ba7f1ef90aad961cf074af15b9ef44a09"  
}
```

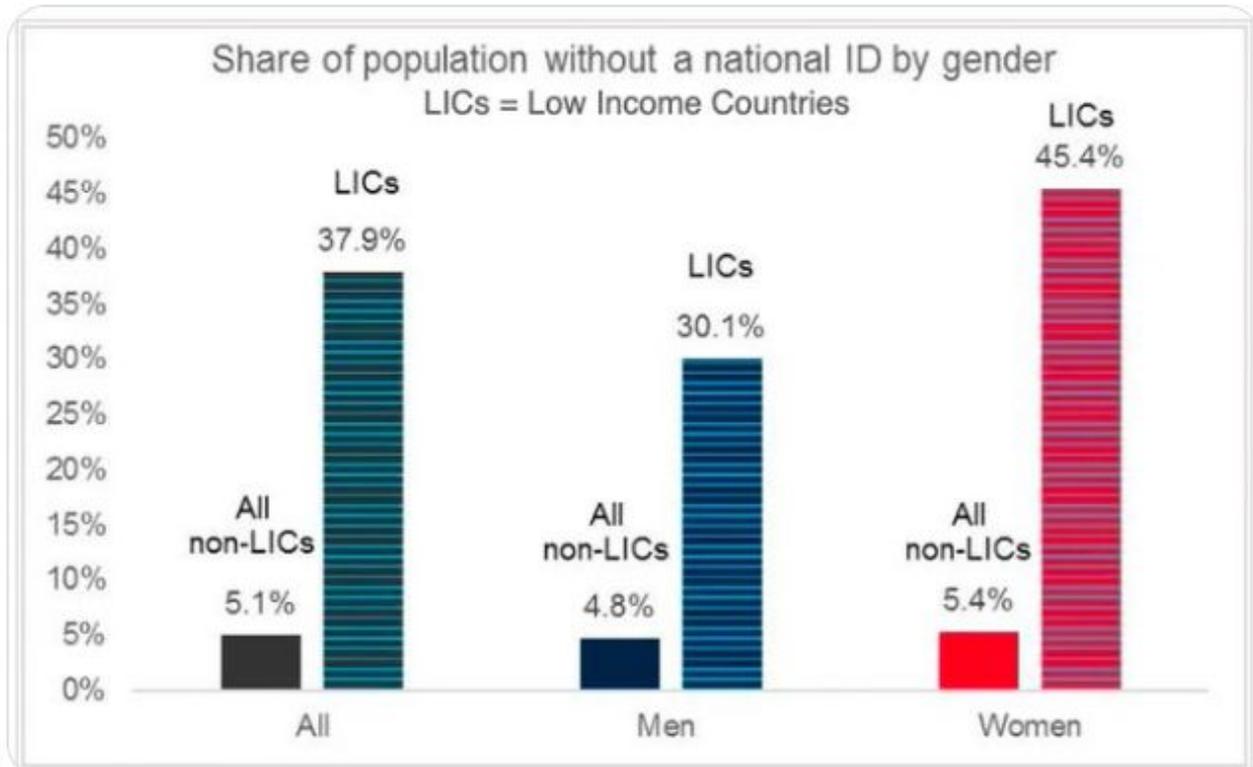
learn how to get NIP05 verified [here](#)



Level39 @level39 · Mar 4

...

13) For the 1.1 billion people who are not able to formally prove their identity, purchasing a personal domain isn't an option. And even if they could, they are unlikely to be able to maintain a domain.





Level39 @level39 · Mar 4



14) Domains are also easily blocked or taken down, especially in authoritarian countries, making them unreliable for identity verification. This has led to concepts such as "unstoppable domains" which many browsers still refuse to recognize.



# What are NIPS? (Hint: +/- 1.5 ounces?)

"Nostr Implementation Possibilities"

## NIPs

NIPs stand for **Nostr Implementation Possibilities**. They exist to document what may be implemented by Nostr-compatible *relay* and *client* software.

- NIP-01: Basic protocol flow description
- NIP-02: Contact List and Petnames
- NIP-03: OpenTimestamps Attestations for Events
- NIP-04: Encrypted Direct Message
- NIP-05: Mapping Nostr keys to DNS-based internet identifiers
- NIP-06: Basic key derivation from mnemonic seed phrase
- NIP-07: `winnow_nostr`: capability for web browsers
- NIP-08: Handling Mentions
- NIP-09: Event Deletion
- NIP-10: Conventions for clients' use of `o` and `p` tags in text events
- NIP-11: Relay Information Document
- NIP-12: Generic Tag Queries
- NIP-13: Proof of Work
- NIP-14: Subject tag in text events.
- NIP-15: End of Stored Events Notice
- NIP-16: Event Treatment
- NIP-19: bech32-encoded entities
- NIP-20: Command Results
- NIP-21: `nostr:` URL scheme
- NIP-22: Event `created_at` Limits
- NIP-23: Long-form Content
- NIP-25: Reactions
- NIP-26: Delegated Event Signing
- NIP-28: Public Chat
- NIP-33: Parameterized Replaceable Events
- NIP-36: Sensitive Content
- NIP-39: External Identities in Profiles
- NIP-40: Expiration Timestamp
- NIP-42: Authentication of clients to relays
- NIP-46: Nostr Connect
- NIP-50: Keywords filter
- NIP-51: Lists
- NIP-56: Reporting
- NIP-57: Lightning Zaps
- NIP-58: Badges
- NIP-65: Relay List Metadata
- NIP-78: Application-specific data



Image:

<https://www.nantucket-ma.gov/2621/Nip-Bottles-of-less-than-or-equal-to-100>

<https://www.marketwatchmag.com/small-spirits-bottle-ban-in-massachusetts-draws-big-response/>

<https://www.masspack.org/>

<https://github.com/nostr-protocol/nips>



**Level39** @level39 · Mar 4



15) Furthermore, an attacker can create imposter identities and use those identities to falsely "verify" captured Nostr identities. Without a standardized method of attestations that users, browsers and machines alike can quickly and easily verify, confusion and fraud ensues.

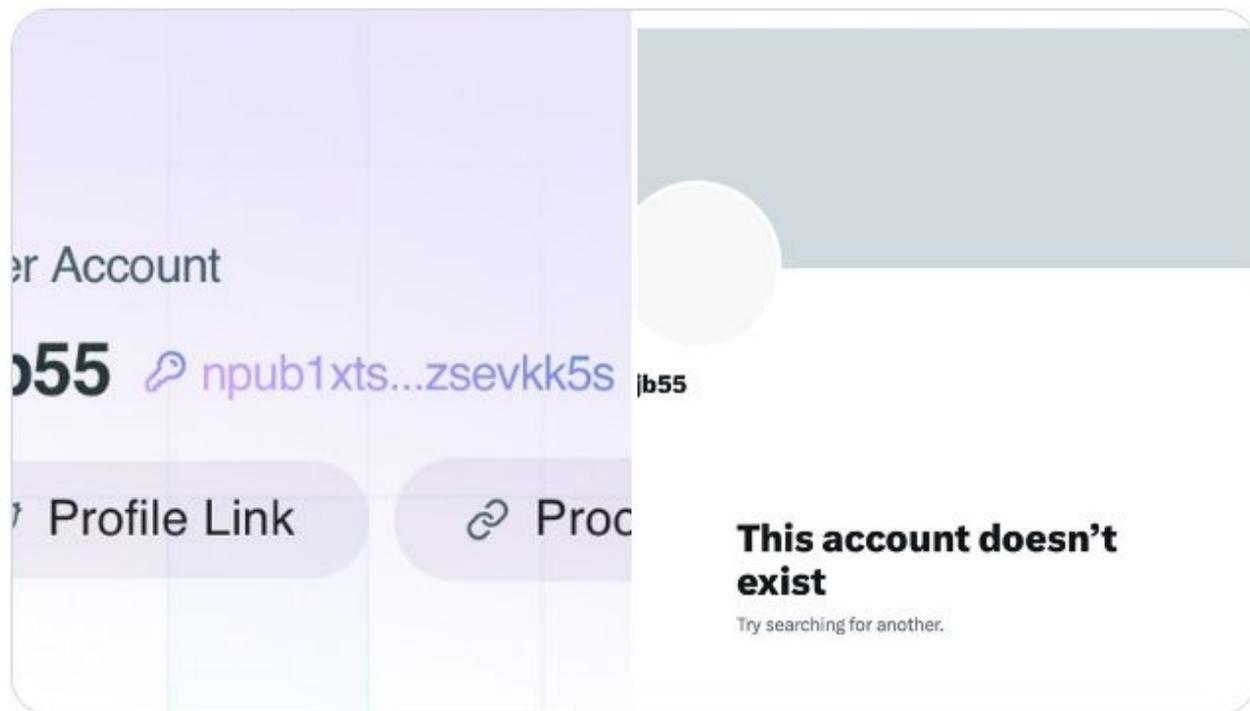




Level39 @level39 · Mar 4



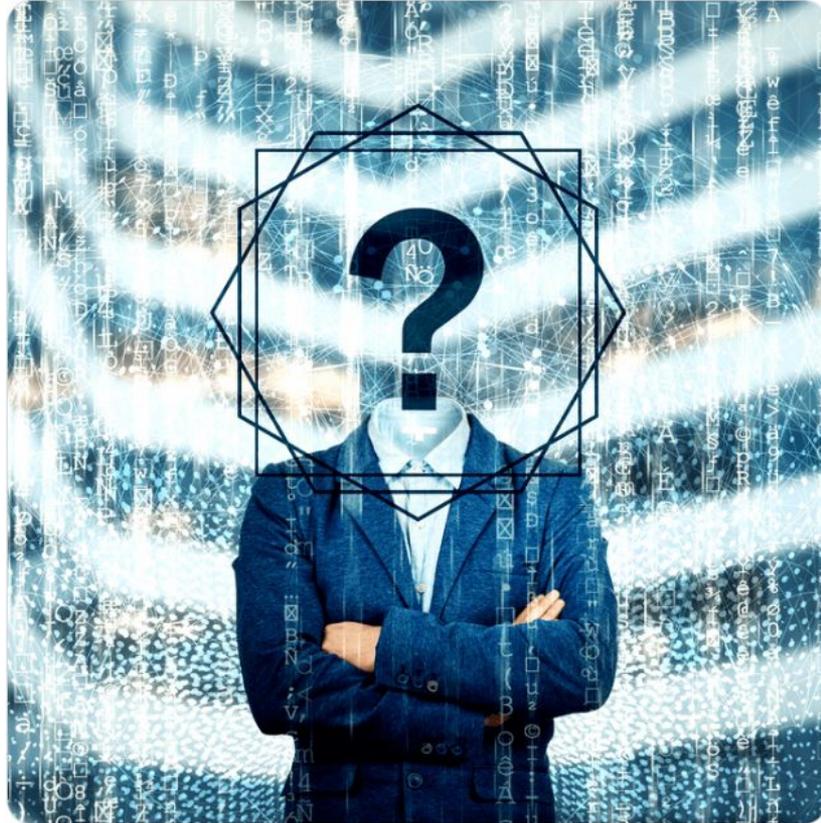
16) Ironically, Nostr dev @jb55 recently lost his Twitter account. If his Nostr key was compromised his imposter would have full control over his Nostr identity and could point his “Proof Link” to a new Twitter account and claim it was his.



39

Level39 @level39 · Mar 4

17) If Nostr cannot be reliably used to manage one's identity, then it will not be a reliable way to communicate in low-income countries and may even be considered an anti-pattern by vendors that are working to integrate with more robust decentralized identity protocols.



39

Level39 @level39 · Mar 4

18) Nostr private keys are almost invariably managed by users without a secure enclave, making those keys easy to mismanage. Even if it were easy to encode Nostr keys into a secure enclave, users would not be able to revoke their keys in the event of theft or loss of devices.

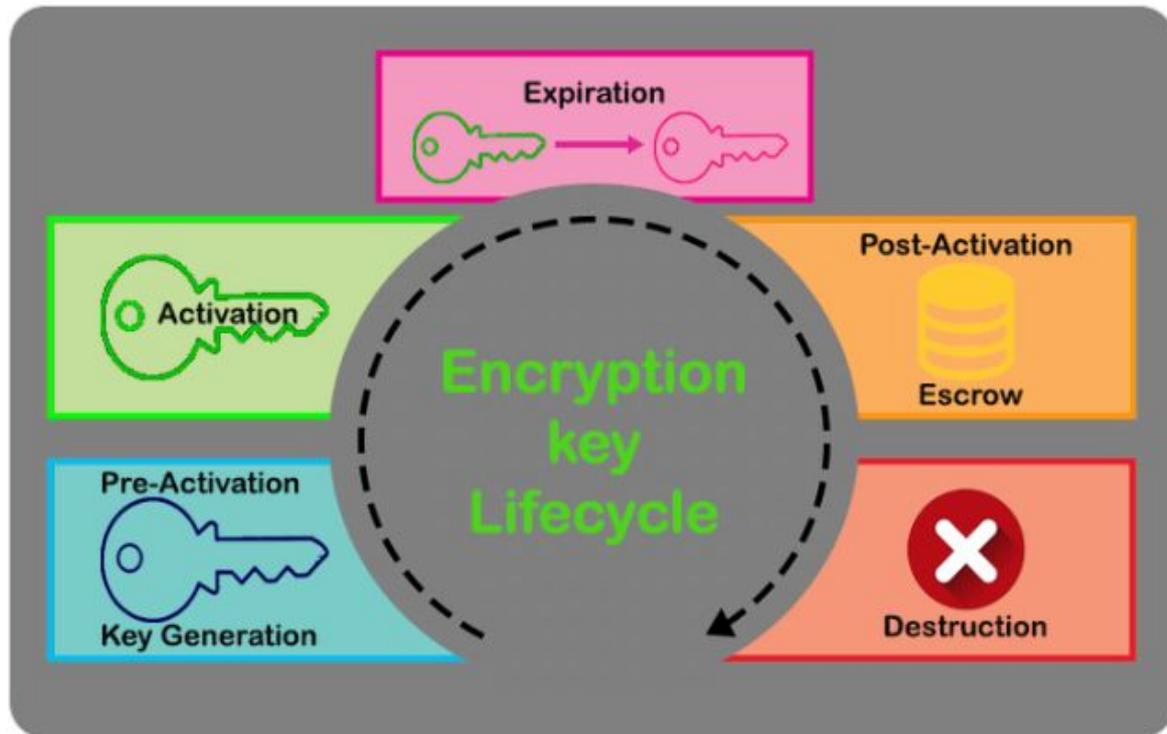




Level39 @level39 · Mar 4

...

19) This is because there is no way to do what is known as "key rolling" on Nostr — a best practice for security. That is, when a key or device becomes compromised or retired, you can't revoke it and keep using your identity. Imagine the following scenario...





**Level39** @level39 · Mar 4



20) Let's say you use Nostr keys to manage a photo album service. 20 years in, an attacker gets access to your keys. With Nostr, you are powerless and can't block the attacker's access. In that sense, Nostr is a protocol for the benefit of relays, not users.

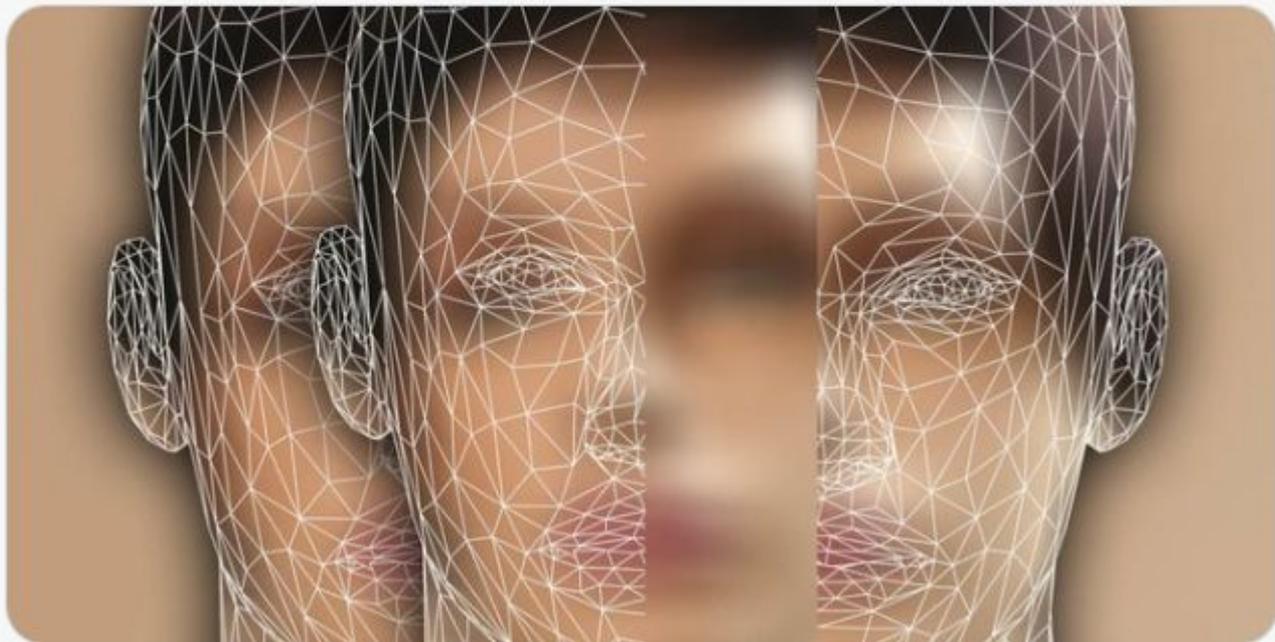




**Level39** @level39 · Mar 4



21) Without regular key rotation and revocation, an attacker could gain access to your Nostr identity, along with any authorized accounts, and spy on you indefinitely without your knowledge. It's good practice to regularly roll and revoke keys, but Nostr doesn't permit this.





Level39 @level39 · Mar 4



22) The good news is that some (imperfect) solutions have been proposed to solve some of these issues in Nostr. For more details, read [@brian\\_trollz's](#) excellent analysis here:



bitcoinmagazine.com

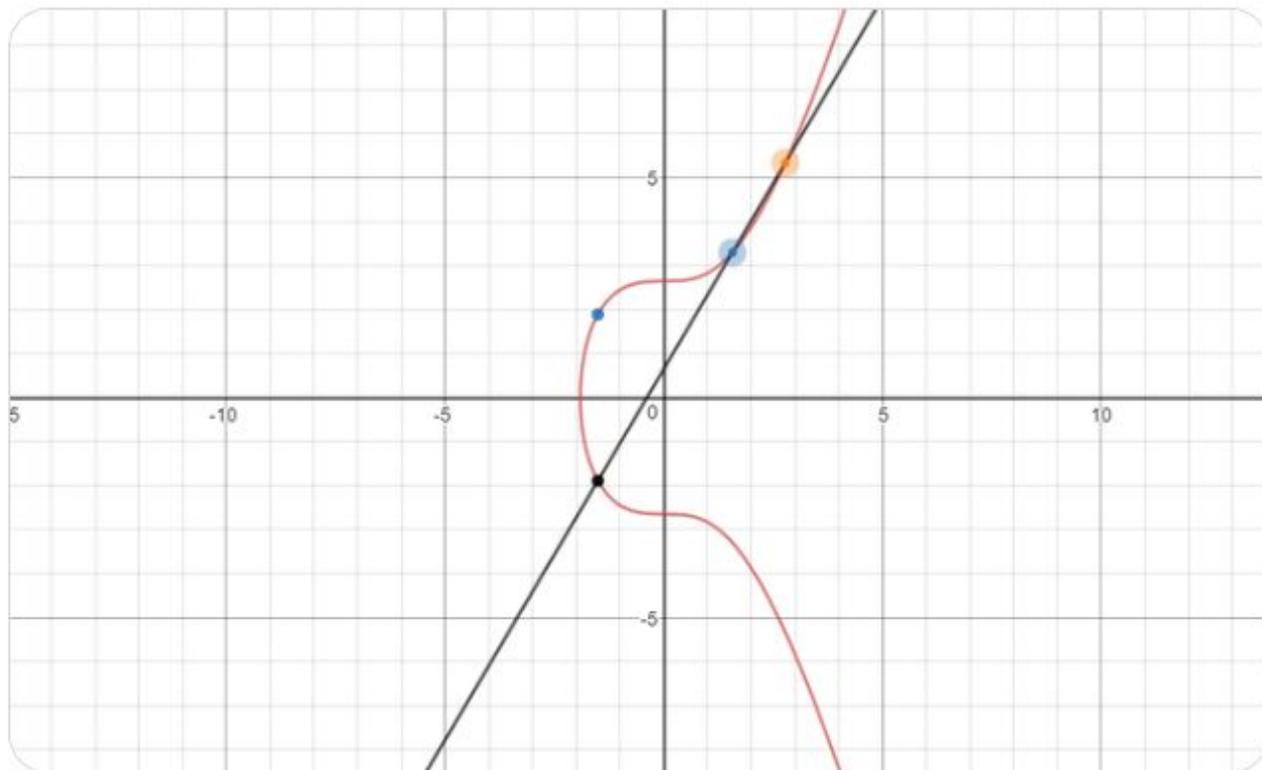
To Become Bitcoin's Go-To Platform, Nostr Will Have To Solve Its K...  
As Bitcoiners turn to Nostr as a censorship-resistant communication platform, user key management problems will arise.



Level39 @level39 · Mar 4

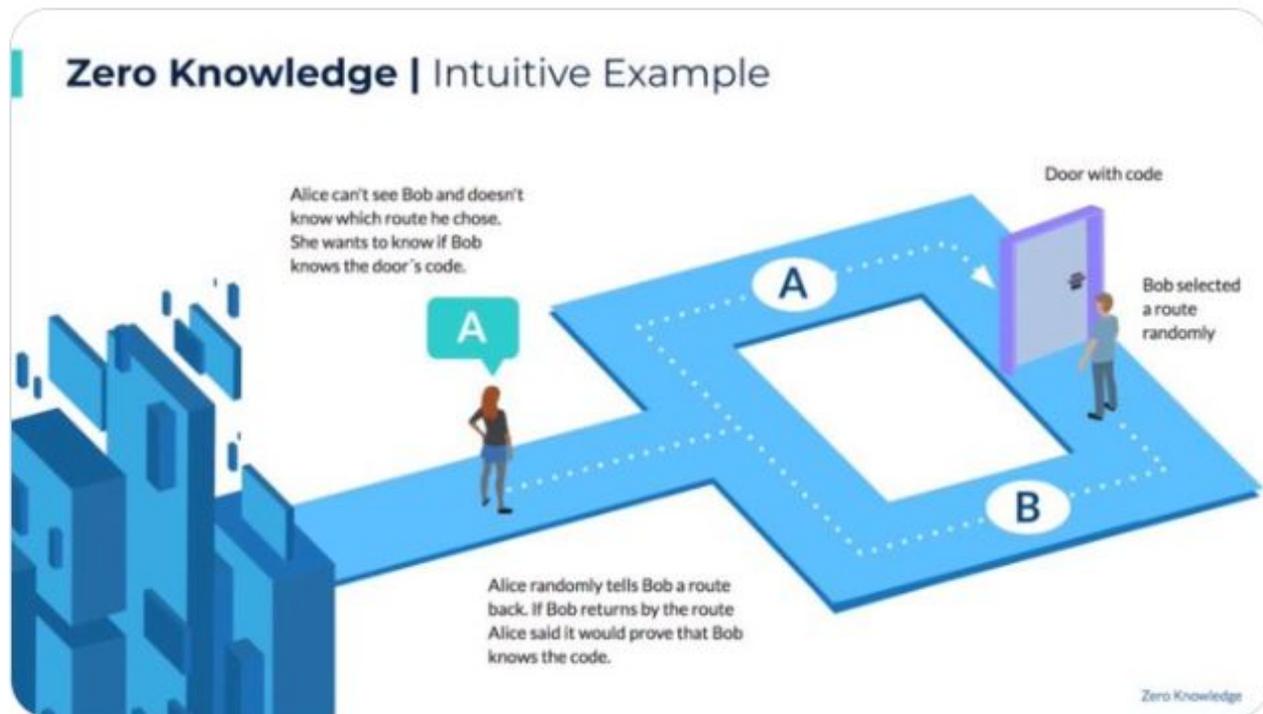


23) Nostr uses a secp256k1—a secure and widely used curve (including for Bitcoin)—to create secure digital signatures and other cryptographic operations. However, it is not performant for all applications.





24) Zero Knowledge Proofs (ZKPs) enable a party to selectively prove data (like your age) without revealing private details—providing a higher degree of privacy and security for personal data. Services with high volume ZKPs will prefer more performant “pairing-friendly” curves.

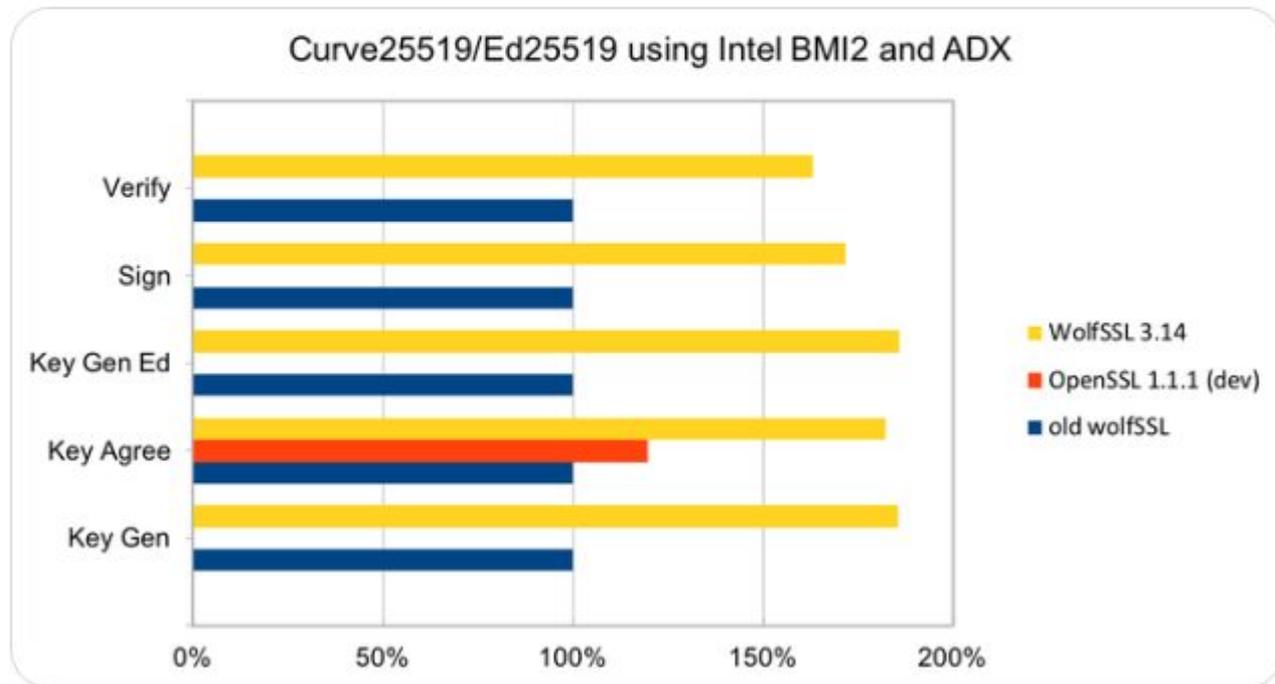




Level39 @level39 · Mar 4



25) For example, GitHub recommends using curves that provide a high level of security and efficiency to implement and use. Curve25519 and Ed25519 are recommended, and are well-suited for use in systems like GitHub where performance is important.

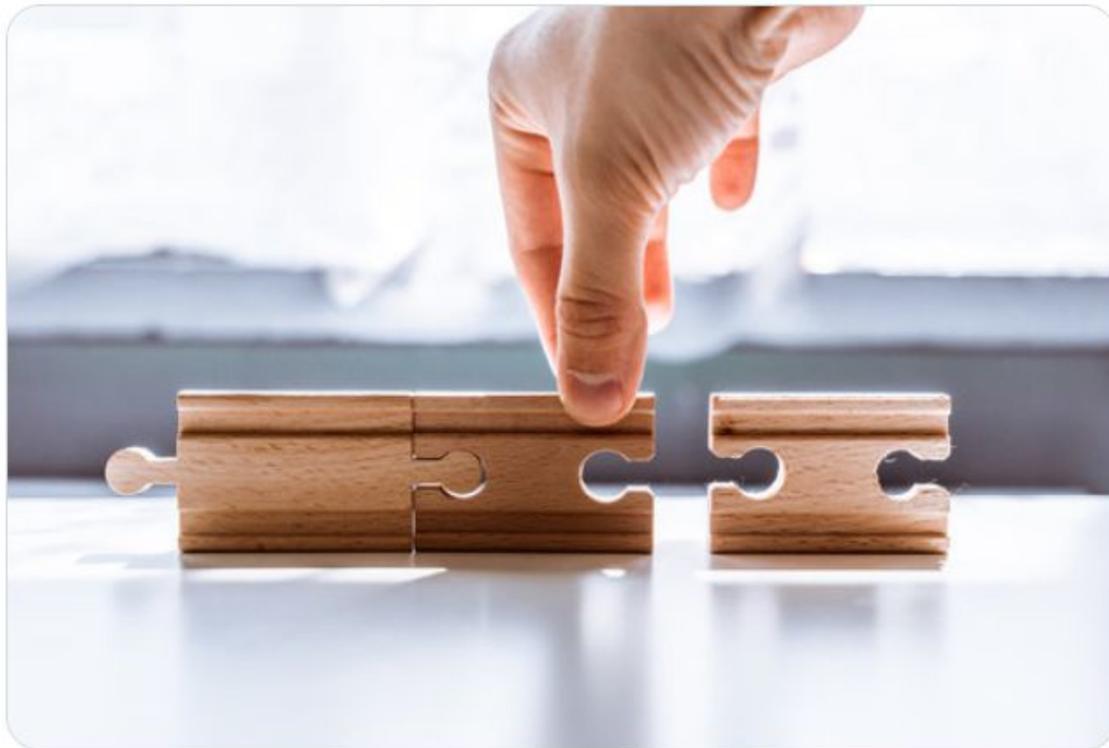




**Level39** @level39 · Mar 4



26) Some applications need additional features and capabilities that are not available with secp256k1. Tying one's identity to secp256k1 may limit users' ability to integrate with other systems and networks, creating potential interoperability issues for users.





**Level39** @level39 · Mar 4



27) If a service requires a more performant elliptic curve at scale, Nostr would lack interoperability with that service. One way to solve these issues would be to use an identity “document” that could robustly list, rotate, revoke and manage keys. Hold onto that thought.





**Level39** @level39 · Mar 4



28) Nostr is a lightweight technology and much like the concerns of the Blocksize Wars its creators understandably want to keep it lightweight. However, the current practice of verifying "out of band" comes with real risks for identity that must either be acknowledged or fixed.

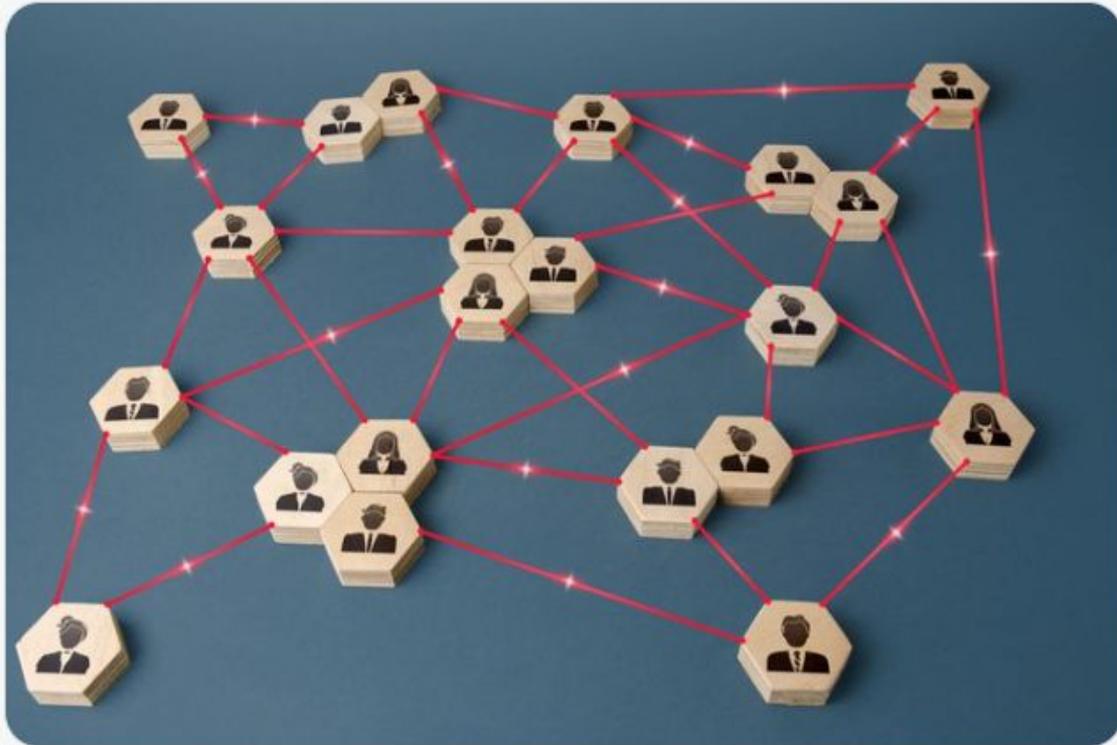




Level39 @level39 · Mar 4



29) Nostr users might not be aware, but a comprehensive decentralized identity solution was developed over the last few years and was recently standardized by @W3C, the main international standards organization for the World Wide Web, founded in 1994 and led by Tim Berners-Lee.





**Level39** @level39 · Mar 4



30) Last year, Berners-Lee promoted Decentralized Identifiers (DIDs) as a new technology-agnostic web standard which permits users to inexpensively own trusted pseudonymous digital identities, without having to rely on centralized protocols. [w3.org/TR/did-core/](https://w3.org/TR/did-core/)

# What is Nostr, the Jack Dorsey-backed social network?

The idea is to create a decentralized social protocol that is independent of apps built on it.

By **Shubham Agarwal**

February 6, 2023



Credit: Damus

# NIP-01

## Basic protocol flow description

draft mandatory author:fiatjaf author:distbit author:scsibug author:kukks author:jb55

This NIP defines the basic protocol that should be implemented by everybody. New NIPs may add new optional (or mandatory) fields and messages and features to the structures and flows described here.

## Events and signatures

Each user has a keypair. Signatures, public key, and encodings are done according to the [Schnorr signatures standard for the curve secp256k1](#).

The only object type that exists is the `event`, which has the following format on the wire:

```
{
  "id": <32-bytes lowercase hex-encoded sha256 of the the serialized event data>,
  "pubkey": <32-bytes lowercase hex-encoded public key of the event creator>,
  "created_at": <unix timestamp in seconds>,
  "kind": <integer>,
  "tags": [
    ["e", <32-bytes hex of the id of another event>, <recommended relay URL>],
    ["p", <32-bytes hex of a pubkey>, <recommended relay URL>],
    ... // other kinds of tags may be included later
  ],
  "content": <arbitrary string>,
  "sig": <64-bytes hex of the signature of the sha256 hash of the serialized event data, which is the same as the "id" fie
}
```

To obtain the `event.id`, we `sha256` the serialized event. The serialization is done over the UTF-8 JSON-serialized string (with no white space or line breaks) of the following structure:

```
[
  0,
  <pubkey, as a (lowercase) hex string>,
  <created_at, as a number>,
  <kind, as a number>,
  <tags, as an array of arrays of non-null strings>,
  <content, as a string>
]
```

# Communication between clients and relays

---

Relays expose a websocket endpoint to which clients can connect.

## From client to relay: sending events and creating subscriptions

Clients can send 3 types of messages, which must be JSON arrays, according to the following patterns:

- `["EVENT", <event JSON as defined above>]`, used to publish events.
- `["REQ", <subscription_id>, <filters JSON>...]`, used to request events and subscribe to new updates.
- `["CLOSE", <subscription_id>]`, used to stop previous subscriptions.

`<subscription_id>` is an arbitrary, non-empty string of max length 64 chars, that should be used to represent a subscription.

`<filters>` is a JSON object that determines what events will be sent in that subscription, it can have the following attributes:

```
{
  "ids": <a list of event ids or prefixes>,
  "authors": <a list of pubkeys or prefixes, the pubkey of an event must be one of these>,
  "kinds": <a list of a kind numbers>,
  "#e": <a list of event ids that are referenced in an "e" tag>,
  "#p": <a list of pubkeys that are referenced in a "p" tag>,
  "since": <an integer unix timestamp, events must be newer than this to pass>,
  "until": <an integer unix timestamp, events must be older than this to pass>,
  "limit": <maximum number of events to be returned in the initial query>
}
```

**Related to some  
questions / discussion**

**Added  
After  
Meeting**

```
func Example() {
    // A keyset created with "tinkey create-keyset --key-template=AES256_GCM". Note
    // that this keyset has the secret key information in cleartext.
    jsonKeyset := `{
        "key": [{
            "keyData": {
                "keyMaterialType":
                    "SYMMETRIC",
                "typeUrl":
                    "type.googleapis.com/google.crypto.tink.AesGcmKey",
                "value":
                    "GiBWyUfGgYk3RTRhj/LIUzSudIWlyjCftCOypTr0jCNSLg=="
            },
            "keyId": 294406504,
            "outputPrefixType": "TINK",
            "status": "ENABLED"
        }],
        "primaryKeyId": 294406504
    }`
}
```

[https://github.com/google/tink/blob/master/go/aead/aead\\_test.go](https://github.com/google/tink/blob/master/go/aead/aead_test.go)

Added  
After  
Meeting

[https://github.com/google/tink/blob/master/go/aead/aead\\_test.go](https://github.com/google/tink/blob/master/go/aead/aead_test.go)

```
// Create a keyset handle from the cleartext keyset in the previous
// step. The keyset handle provides abstract access to the underlying keyset to
// limit the exposure of accessing the raw key material. WARNING: In practice,
// it is unlikely you will want to use an insecurecleartextkeyset, as it implies
// that your key material is passed in cleartext, which is a security risk.
// Consider encrypting it with a remote key in Cloud KMS, AWS KMS or HashiCorp Vault.
// See https://github.com/google/tink/blob/master/docs/GOLANG-HOWTO.md#storing-and-loading-existing
keysetHandle, err := insecurecleartextkeyset.Read(
    keyset.NewReader(bytes.NewBufferString(jsonKeyset)))
if err != nil {
    log.Fatal(err)
}

// Retrieve the AEAD primitive we want to use from the keyset handle.
primitive, err := aead.New(keysetHandle)
if err != nil {
    log.Fatal(err)
}

// Use the primitive to encrypt a message. In this case the primary key of the
// keyset will be used (which is also the only key in this example).
plaintext := []byte("message")
associatedData := []byte("associated data")
ciphertext, err := primitive.Encrypt(plaintext, associatedData)
if err != nil {
    log.Fatal(err)
}

// Use the primitive to decrypt the message. Decrypt finds the correct key in
// the keyset and decrypts the ciphertext. If no key is found or decryption
// fails, it returns an error.
decrypted, err := primitive.Decrypt(ciphertext, associatedData)
if err != nil {
    log.Fatal(err)
}

fmt.Println(string(decrypted))
// Output: message
```

Added  
After  
Meeting

# Tinkey

<https://github.com/google/tink/blob/master/docs/TINKEY.md>

<https://github.com/google/tink/blob/master/docs/GOLANG-HOWTO.md>

Added  
After  
Meeting

This utility allows generating and manipulating Tink keysets. It can encrypt or decrypt keysets with master keys residing in a remote key management service (KMS). Out of the box it supports AWS KMS and Google Cloud KMS. Adding support for other KMS is easy, and doesn't require modifying Tinkey.

Tinkey requires Java 8 or later to run.

## Generating new keys and keysets

To take advantage of key rotation and other key management features, you usually do not work with single keys, but with keysets. Keysets are just sets of keys with some additional parameters and metadata.

Internally Tink stores keysets as Protocol Buffers, but you can work with keysets via a wrapper called a keyset handle. You can generate a new keyset and obtain its handle using a KeyTemplate. KeysetHandle objects enforce certain restrictions that prevent accidental leakage of the sensitive key material.

"Using crypto in your application shouldn't have to feel like juggling chainsaws in the dark."



# Tink

## Primitives and Interfaces

### I want to encrypt data

### Keys

### Keysets

### Goals

```
# A keyset created with "tinkey create-keyset --key-template=AES256_GCM". Note
# that this keyset has the secret key information in cleartext.
keyset = r"""{
  "key": [{
    "keyData": {
      "keyMaterialType":
        "SYMMETRIC",
      "typeUrl":
        "type.googleapis.com/google.crypto.tink.AesGcmKey",
      "value":
        "GiBWYUFGyYk3RTRhj/LIUzSudIW1yjCftC0ypTr0jCNSLg=="
    },
    "keyId": 294406504,
    "outputPrefixType": "TINK",
    "status": "ENABLED"
  }],
  "primaryKeyId": 294406504
}"""
```

See <https://github.com/google/tink>

## Tink

**NOTE: Tink is moving!**

As part of our roadmap we are splitting Tink into [multiple GitHub repositories](#) that will be hosted at [github.com/tink-crypto](https://github.com/tink-crypto) and will be independently versioned.

Roughly, we are going to create one repository per language, library extension such as KMS (except Python), and tools.

A few important highlights:

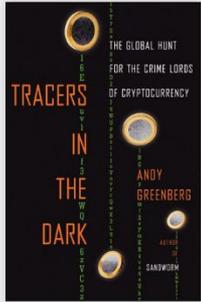
- The migration will be done gradually over the course of 2023 with a new release from each of the new repositories. Releases will be announced in our [mailing list](#).
- We will keep updating each implementation/tool in [github.com/google/tink](https://github.com/google/tink) for a specified amount of time; migrated implementations/tools will eventually stop being updated on [github.com/google/tink](https://github.com/google/tink). The support window depends on the specific implementation, as shown in the table below.
- New issues and pull requests should be created in the new repos.

Tink Python	<a href="https://github.com/tink-crypto/tink-py">tink-crypto/tink-py</a>	Not started (expected Q2 2023)	TBA
Tink Go	<a href="https://github.com/tink-crypto/tink-go">tink-crypto/tink-go</a>	Not started (expected Q3 2023)	TBA
Tink Go AWS KMS extension	<a href="https://github.com/tink-crypto/tink-go-awskms">tink-crypto/tink-go-awskms</a>	Not started (expected Q3 2023)	TBA
Tink Go Google Cloud KMS extension	<a href="https://github.com/tink-crypto/tink-go-gcpkms">tink-crypto/tink-go-gcpkms</a>	Not started (expected Q3 2023)	TBA

**Added  
After  
Meeting**

# What is Crypto, Bro?

★ Earn 10 points on this purchase!



 Add to Bookshelf

[Look Inside](#)

## Tracers in the Dark

THE GLOBAL HUNT FOR THE CRIME LORDS OF CRYPTOCURRENCY

By Andy Greenberg

Category: [Biography & Memoir](#)

Paperback +

Hardcover -

Hardcover \$32.50

Nov 15, 2022 | ISBN  
9780385548090

[ADD TO CART](#)

Buy from Other Retailers:

[Amazon](#)

[Barnes & Noble](#)

[Books A Million](#)

[Bookshop.org](#)

[Hudson Booksellers](#)

[Powell's](#)

[Target](#)

[Walmart](#)

Ebook +

Audio +

Join [Reader Rewards](#) to earn points on this purchase. 

### ABOUT TRACERS IN THE DARK

From the award-winning author of *Sandworm* comes the propulsive story of a new breed of investigators who have cracked the Bitcoin blockchain, exposing once-anonymous realms of money, drugs, and violence. "I love the book... It reads like a thriller... These stories are amazing." (Michael Lewis)

"Over the last decade, a single innovation has massively fueled digital black markets: cryptocurrency. Crime lords inhabiting lawless corners of the internet have operated more freely—whether in drug dealing, money laundering, or human trafficking—than their analog counterparts could have ever dreamed of. By transacting not in dollars or pounds but in currencies with anonymous ledgers, overseen by no government, beholden to no bankers ..."

Source:

<https://www.penguinrandomhouse.com/books/690603/tracers-in-the-dark-by-andy-greenberg/>

Added  
After  
Meeting

# What about hardware devices?



fiatjaf @fiatjaf · 6h

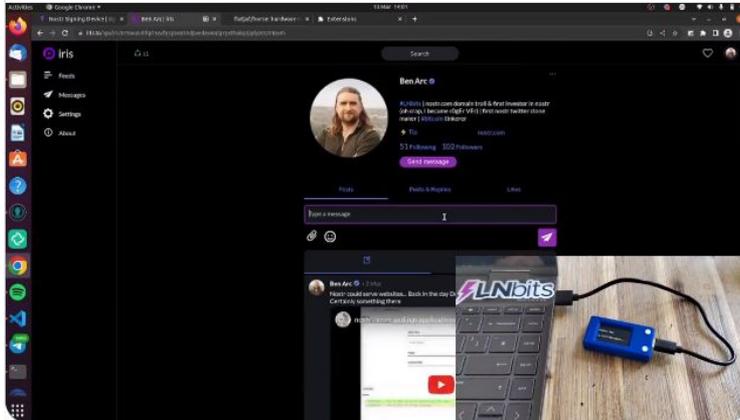
NIP-07 signing from a hardware device is happening.

Ben Arc 🇨🇦 🇺🇸 ⚡ @arcbtc · Mar 13

Introducing the worlds first Nostr Signing Device!

🔑 Protect those keys 🔑

[Show this thread](#)



- \*\$10 to make
- \*Off the shelf hardware
- \*Web-installer for easy build
- \*Compatible with most clients



<https://twitter.com/i/lists/1605941759061663744>

<https://github.com/lnbits/nostr-signing-device>

Added  
After  
Meeting



# Hardware Signing Device?

AliExpress

lilygo Official Store

+ Follow

Top Brand 98.2% Positi... 17058 Followers

I'm shopping for...

On AliExpress

In this store



me Products v Anniversary Sale Top Selling Feedback Brand Story



LILYGO® TTGO T-Display 1.14 Inch LCD Control Board ESP32 Wireless Module  
WiFi Bluetooth Low Power Consumption Development Board

US \$5 off every US \$30 (max US \$15) | Extra 2% off

★★★★★ 5.0 v 40 Reviews 352 orders

Exclusive first order price

US \$10.48

~~US \$14.98~~ 30% off

Welcome Deal

Store Discount: Buy 2 get 5% off v

[Get coupons](#)

Color: 4MB CH9102F

16MB CH340K

16MB CH9102F

16MB 9102 With Case

16MB Solde With Case

4MB CH340K

4MB CH9102F

4MB 9102 With Case

Quantity

Added  
After  
Meeting

<https://www.aliexpress.us/item/2251832862647579.html>

## Mastodon loses 30% of its active users after Twitter fears die down

Mastodon swept up a bunch of Twitter's users last year, but they're not sticking around



Callum Bains  
Jan 9



(Credit: Mastodon)

### → The Shortcut Skinny: Mastodon extinction?

- 🐼 Mastodon is having trouble keeping users
- 🧡 Many disgruntled Twitter users jumped to Mastodon last year
- 📄 But new data shows they're not sticking around on the platform
- 😬 Possibly because it's too confusing to use

"As first reported by The Guardian, Mastodon's latest batch of user data shows the number of active users on the social media platform has fallen dramatically over the last month. In early December, Mastodon reportedly attracted over 2.2 million active users. By the end of the first week of January, that figure had dropped to just over 1.7 million - a fall of 30%"

<https://www.theshortcut.com/p/twitter-death-mastodon-loses-active-users>

This link was pointed to by a helpful attendee in the meeting's chat.

Added  
After  
Meeting

# Elon Musk drove more than a million people to Mastodon - but many aren't sticking around

More than 130,000 people were joining the new independent social media network a day in November. So why hasn't it taken off?

● Follow our Australia news live blog for the latest updates

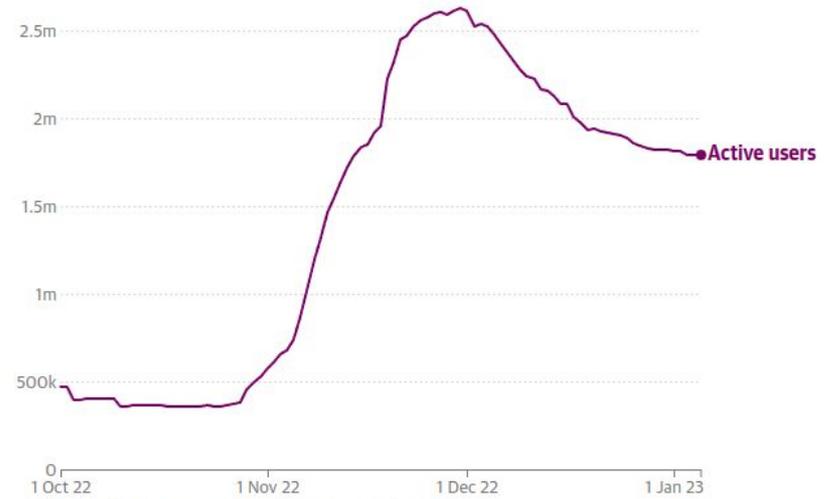


📷 The open-source social network Mastodon grew rapidly during Elon Musk's initial takeover of Twitter but active users are now steadily declining. Photograph: Rafael Henrique/SOPA Images/Rex/Shutterstock

The number of active users on the Mastodon social network has dropped more than 30% since the peak and is continuing a slow decline, according to the [latest data posted on its website](#). There were about 1.8 million active users in the first week of January, down from over 2.5 million in early December.

## Active users on Mastodon servers

Showing the total number of users that have logged in over the previous 30 days. Only showing days for which there is complete data.



Guardian graphic | Source: Joinmastodon.org, Wayback Machine

<https://www.theguardian.com/news/datablog/2023/jan/08/elon-musk-drove-more-than-a-million-people-to-mastodon-but-many-arent-sticking-around>

Added  
After  
Meeting

# *What Is Mastodon and Why Are People Leaving Twitter for It?*

Since Elon Musk took ownership of Twitter, some of its users have migrated to Mastodon, an alternative social platform.

November 7, 2022 - <https://www.nytimes.com/2022/11/07/technology/mastodon-twitter-elon-musk.html>

INFINITE SCROLL

## WHAT FLEEING TWITTER USERS WILL—AND WON'T—FIND ON MASTODON

*The burgeoning social network is “designed to be against virality,” as one user put it. Can it be the future of social media?*



By Kyle Chayka  
November 22, 2022

Added  
After  
Meeting

The New Yorker, November 22, 2022

<https://www.newyorker.com/culture/infinite-scroll/what-fleeing-twitter-users-will-and-wont-find-on-mastodon>



**Keith Mukai**  
@KeithMukai

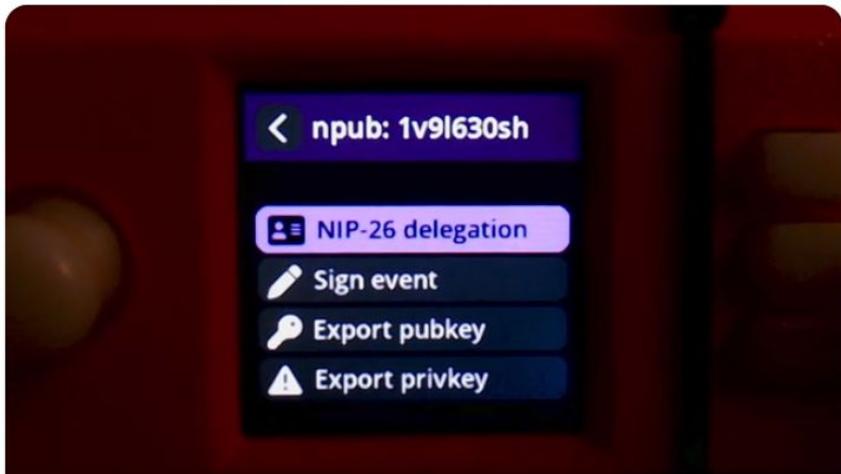


Finally! @SeedSigner + Nostr Demo 3: NIP-26 delegation!!

A completely airgapped key uses SeedSigner to authorize a delegatee to sign on its behalf.

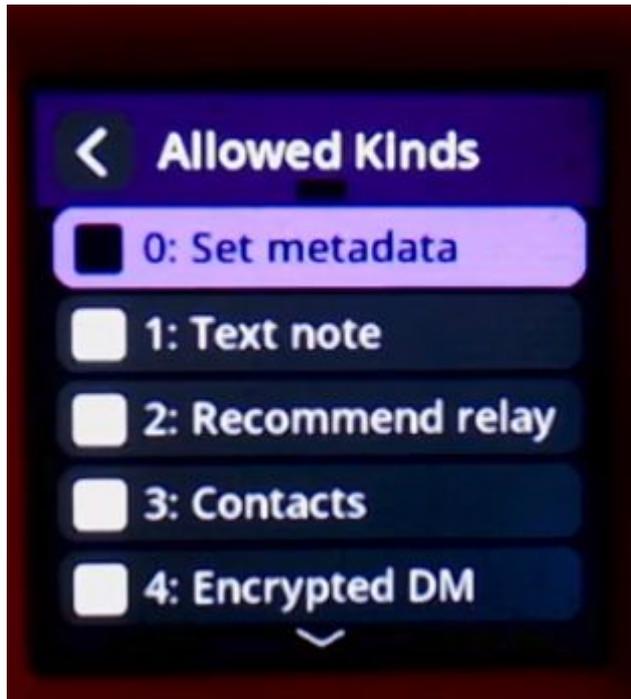
We then create a new event, sign it with the delegatee's key, and successfully publish it!

#nip26



youtube.com

SeedSigner + Nostr Demo 3: NIP-26 Delegation(!), pt1  
Full end-to-end airgapped NIP-26 delegation. The "AirGappedKeith" key authorizes the npub1mypr0xy key to sign on its behalf. And then the delegatee...



<https://twitter.com/KeithMukai/status/1620240140177408000>





# Encrypting things in NOSTR

The event kind system was expanded quite substantially from that original NIP. There is an event type for encrypted direct messages, establishing a shared key by combining the sender's private key with the receiver's public key, which results in the same key you would get by combining the sender's public key with the receiver's private key (this is how [BIP 47](#) and Silent Payments work). There are also types for replaceable events and ephemeral events. In the case of a replaceable event (obviously), they are designed so that the original creator of the event can sign a new one to replace the old one. Relay servers following the specification will automatically drop the

<https://bitcoinmagazine.com/technical/what-makes-nostr-a-different-social-platform>





# NOSTR: Encrypt and Decrypt

## Nostr.Crypto.AES256CBC

Algorithm that encrypts and decrypts direct messages

## Summary

## Functions

`decrypt(message, seckey, pubkey)`

`encrypt(message, seckey, pubkey)`

```
decrypt(message, seckey, pubkey) </>
```

```
@spec decrypt(String.t(), K256.Schnorr.signing_key(), K256.Schnorr.verifying_key()) ::  
  {:ok, String.t()} | {:error, atom() | String.t()}
```

```
encrypt(message, seckey, pubkey) </>
```

```
@spec encrypt(  
  String.t(),  
  K256.Schnorr.signing_key() | <<_:256>>,  
  K256.Schnorr.verifying_key() | <<_:256>>  
) :: String.t()
```

<https://hexdocs.pm/nostr/Nostr.Crypto.AES256CBC.html>



# Make them presentable ...

<https://github.com/nostr-protocol/nip/blob/master/19.md>

## NIP-19

---

### bech32-encoded entities

---

draft optional author:jb55 author:fiatjaf author:Semisol

This NIP standardizes bech32-formatted strings that can be used to display keys, ids and other information in clients. These formats are not meant to be used anywhere in the core protocol, they are only meant for displaying to users, copy-pasting, sharing, rendering QR codes and inputting data.

It is recommended that ids and keys are stored in either hex or binary format, since these formats are closer to what must actually be used the core protocol.

### Bare keys and ids

---

To prevent confusion and mixing between private keys, public keys and event ids, which are all 32 byte strings. bech32-(not-m) encoding with different prefixes can be used for each of these entities.

These are the possible bech32 prefixes:

- `npub` : public keys
- `nsec` : private keys
- `note` : note ids

Example: the hex public key `3bf0c63fcb93463407af97a5e5ee64fa883d107ef9e558472c4eb9aaafafa459d` translates to `npub180cgv07tjdrngpa0j7j7tmnyl2yr6yr7l8j4s3evf6u64th6gkwsyjh6w6` .



# Learn More

reddit r/nostr Search Reddit

nostr join

Posts Spec

Create Post

Hot New Top

PINNED BY MODERATORS

63 Posted by u/melvincarvalho 1 year ago

nostr: a truly censorship-resistant alternative to Twitter that has a chance of working

github.com/fiatjaf

9 Comments Award Share Save

32 Posted by u/melvincarvalho 1 year ago

GitHub - aljazeera/awesome-nostr: A curated list of nostr projects and resources

github.com/aljazeera

3 Comments Award Share Save

2 Posted by u/undertheradar48 5 hours ago

Nostrica Discussion Thread

stacker.news/items/...

## Nostr Gateway

Nostr Gateway

The Nostr gateway is an effort to pull Nostr data from relays around the nostrsphere into HTML pages for the consumption of the unnostrinitiated.

Paste a Nostr event key or a public key here in NIP-19 format:

Go

What is Nostr?

A decentralized network based on cryptographic keyspairs and that is not peer-to-peer, it is super simple and scalable and therefore has a chance of working.

Read more at <https://github.com/nostr-protocol/nostr> see a giant list of all things, good and bad, made with Nostr, at [nostr.net](https://nostr.net).

You can also reach us at our [Telegram group](#) while we don't have a decent group chat application fully working on Nostr.

Contribute to this site!

You can find the source code for this gateway website at <https://github.com/fiatjaf/nostr-gateway> while we haven't finished implementing Git over Nostr yet.

The Nostr Gateway is an effort to pull Nostr data from relays around the nostrsphere into HTML pages for the consumption of the unnostrinitiated. It is built using Node.js and Docker, and is designed to be easy to deploy and run on any machine.

<https://github.com/fiatjaf/nostr-gateway>

Added  
After  
Meeting