GNU Anastasis

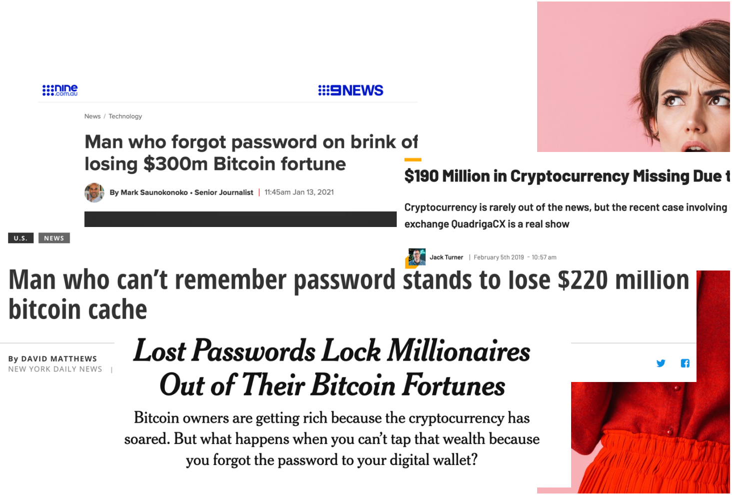
Nullcon 2022, Berlin

funded by

anastasis.lu
anastasis-sarl@twitter

Christian Grothoff
grothoff@anastasis.lu

The Problem Illustrated



9NEWS

News / Technology

Man who forgot password on brink of losing $300m Bitcoin fortune

By Mark Saunokonoko • Senior Journalist | 11:45am Jan 13, 2021

$190 Million in Cryptocurrency Missing Due t

Cryptocurrency is rarely out of the news, but the recent case involving exchange QuadrigaCX is a real show

Jack Turner | February 5th 2019 - 10:57 am

U.S. NEWS

Man who can't remember password stands to lose $220 million bitcoin cache

By DAVID MATTHEWS
NEW YORK DAILY NEWS

Lost Passwords Lock Millionaires Out of Their Bitcoin Fortunes

Bitcoin owners are getting rich because the cryptocurrency has soared. But what happens when you can't tap that wealth because you forgot the password to your digital wallet?

Problem: Availability (1/3)

If you give one person a secret, it may get lost.

Problem: Availability (1/3)

If you give one person a secret, it may get lost.

$\Rightarrow$ So give it to more than one person!

Problem: Confidentiality (2/3)

If you give many entities a secret, it may get disclosed.

Problem: Confidentiality (2/3)

If you give many entities a secret, it may get disclosed.

$\Rightarrow$ So give them only a key share!

Problem: Scalability (3/3)

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

combinations to consider.

Problem: Scalability (3/3)

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

combinations to consider.

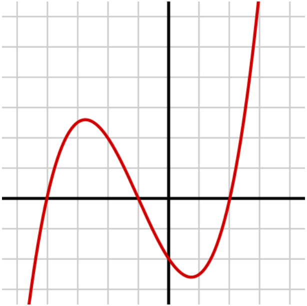
$\Rightarrow$ Use polynominals!

Polynominals

A polynominal of degree k is fully determined by $k+1$ data points

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k),$$

where no two x_j may be identical.



Lagrange Interpolation

The interpolation polynominal in the Lagrange form is:

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

where

$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x}{(x} \tag{2}$$

for $0 \le j \le k$.

Practical Considerations

- Our secrets will typically be integers. Calculations with floating points are *messy*.

$\Rightarrow$ Use finite field arithmetic, not $\mathbb{R}$.

Real world scalability

n / k	1	2	3	4	5	6
1	1	2	3	4	5	6
2		1	3	6	10	15
3			1	4	10	20
4				1	5	15
5					1	6
6						1

Other values:

- $\binom{10}{5} = 252$
- $\binom{20}{10} = 184756$
- $\binom{30}{15} = 155117520$
- $\binom{100}{50} \approx 10^{29}$

Scalability Problem?

How many people do you have to share your secrets with?

How many people realistically participate in recovery?

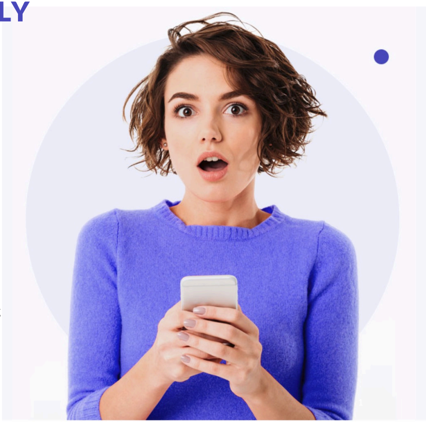
THE PROBLEM TECHNICALLY

Confidentiality requires only consumer is in control of key material. Or in other words, nobody can access your password or secret key.

Consumers are unable to simultaneously ensure confidentiality and availability of keys.

Cryptographic key-splitting solutions so far are not usable.

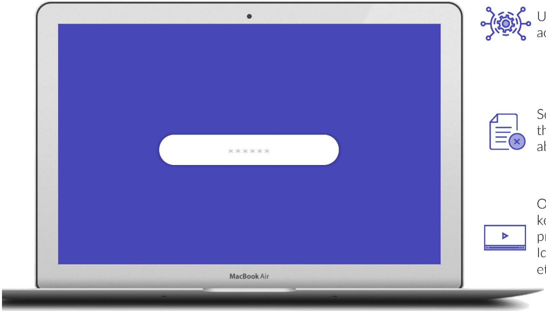
Regulation[1] forces European e-money issuers using electronic wallets to enable consumers to always recover their electronic funds (i.e. if devices are lost).



[1] According to ECB

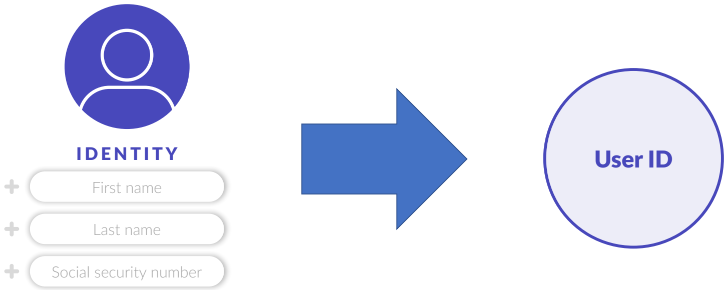
WHAT IS ANASTASIS?

ANASTASIS IS A SECRET/KEY RECOVERY SERVICE WITH FREE & OPEN SOURCE SOFTWARE TO BACK-UP YOUR SECRET WITHOUT DEPENDING ON ANY 3rd PARTY



- Users split their secret keys across multiple service providers
- Service providers learn nothing about the user, except possibly some details about how to authenticate the user
- Only the authorized user can recover the key by following standard authentication procedures (SMS TAN, Video-Identification, Security Question, eMail, etc.)

Preliminaries



Adversary Model

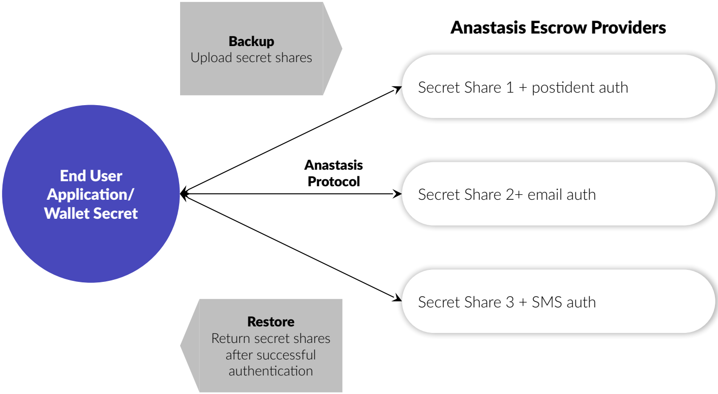
Weak adversary
(does not know the user's identifier)

- Can link requests with the same identifier
- Can learn authentication data (only during authentication)

Strong adversary
(knows the user's identifier)

- Can see if user has account
- Can read recovery information and try to authenticate
- Can get core secret, if
  - providers collude
  - or can solve the authentication

Overview


Backup
Upload secret shares
Anastasis Escrow Providers
Secret Share 1 + postident auth
End User Application/ Wallet Secret
Anastasis Protocol
Secret Share 2+ email auth
Secret Share 3 + SMS auth
Restore
Return secret shares after successful authentication

Simplified Process Flow

Step 1: Core Secret

Simplified Process Flow

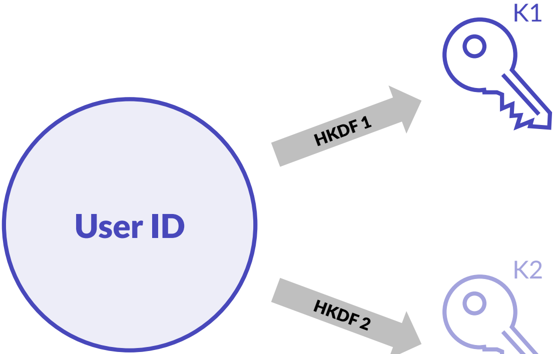
Step 2: Split Core Secret

Simplified Process Flow

Step 3: User Identification

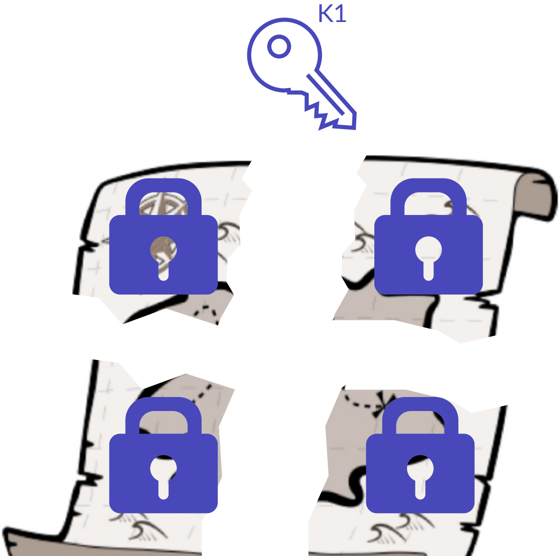
Simplified Process Flow

Step 4: Key Derivation



Simplified Process Flow

Step 5: Encrypt Parts



Simplified Process Flow

Step 6: Add Truth

+ H (answer to security question)

+ T-OTP secret

+ Phone number

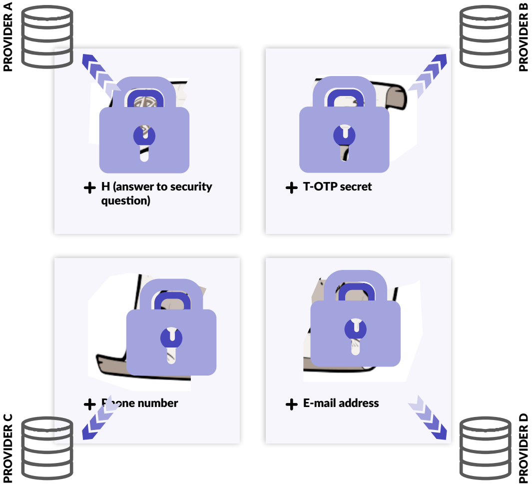
+ E-mail address

Simplified Process Flow

Step 7: Encrypt Truth

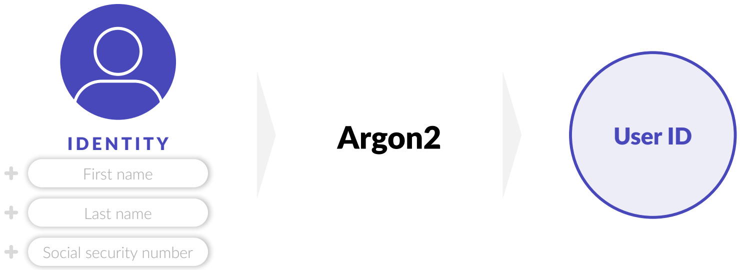
Simplified Process Flow

Step 8: Store Data



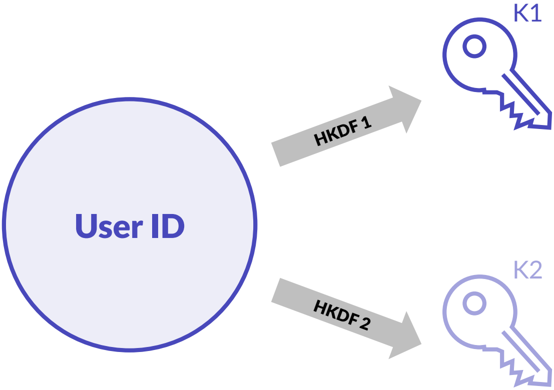
Simplified Process Flow

Step 9: User Identification



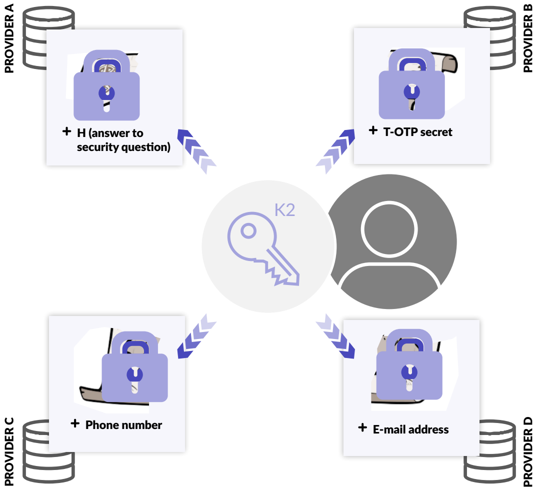
Simplified Process Flow

Step 10: Key Derivation



Simplified Process Flow

Step 11: Provide Key

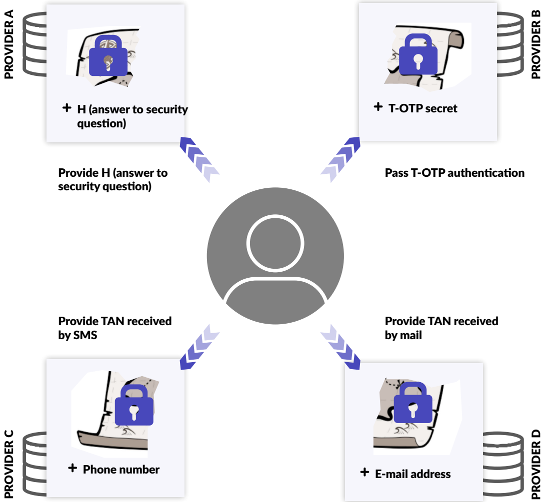


Simplified Process Flow

Step 12: Decrypt Truth

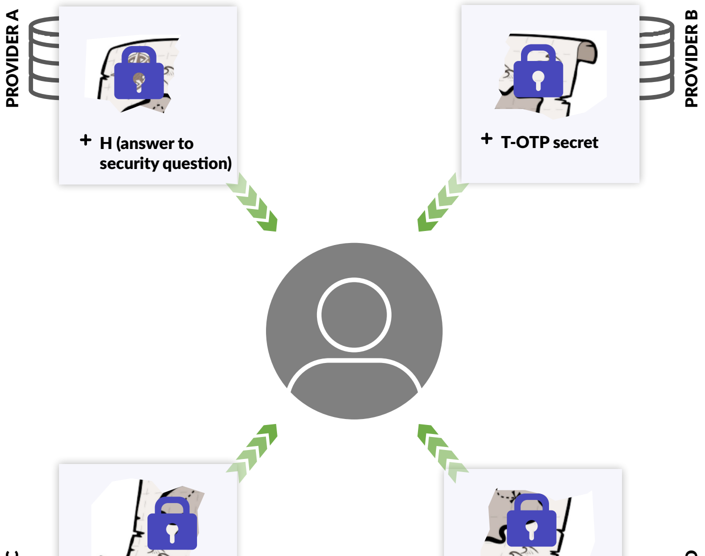
Simplified Process Flow

Step 13: Authentication



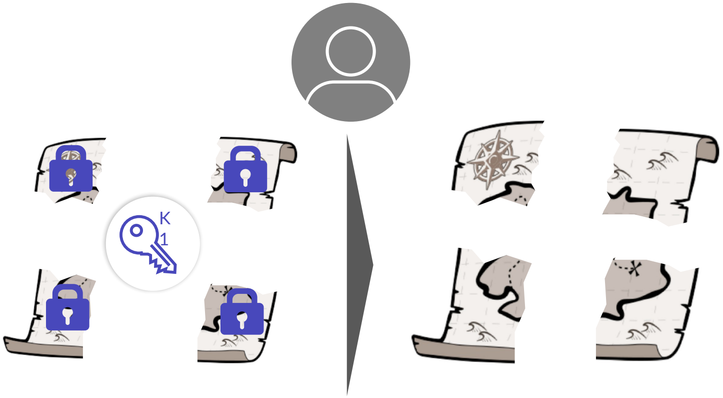
Simplified Process Flow

Step 14: Receive Parts



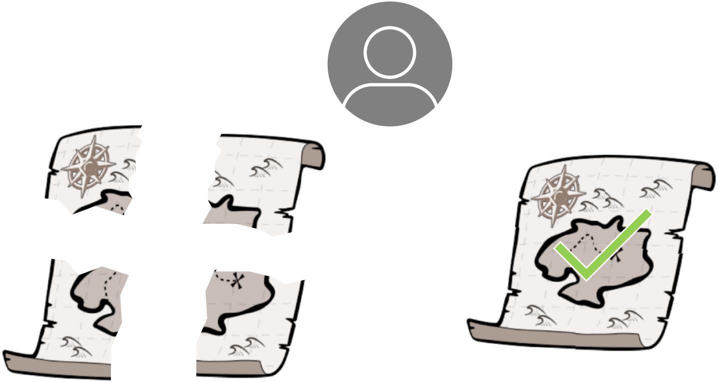
Simplified Process Flow

Step 15: Decrypt Parts



Simplified Process Flow

Step 16: Reassembly



Simplifications

The previous illustrations make various simplifications

- Policies to allow more flexible splitting than 4/4
- Recovery document to remember policies and providers
- Distinction between core secret and master secret
- Provider salts
- Payment processing
- Anti-DoS provisions in protocol / request limits
- Versioning
- Liability limitations

Demonstration

Demo.

Software architecture overview

https://git.taler.net/anastasis.git

Anastasis is a protocol.

The software consists of three components:

anastasis	Backend and client libraries (C)
anastasis-gtk	Gtk+ front-end (C)
anastasis-ts	Alternative front-end (TS)

Major dependencies include:

GNU Taler	Privacy-preserving payments (C/TS)
Postgres	Backend database (C)
libeufin	Alternative access to banking infrastructure (Kotlin)
GNUnet	Various utility functions (C)
GNU MHD	HTTP server library (C)

Binary installation instructions

https://docs.anastasis.lu/

Debian 11:

```
# echo 'deb https://deb.taler.net/apt/debian/ bullseye main'\
  > /etc/apt/sources.list/taler.list
# wget -O - https://taler.net/taler-systems.gpg.key |\
  apt-key add -
# apt update
# apt install anastasis-gtk
```

Ubuntu 20.04:

```
# echo 'deb https://deb.taler.net/apt/ubuntu/ focal-fossa main'\
  > /etc/apt/sources.list/taler.list
# wget -O - https://taler.net/taler-systems.gpg.key |\
  apt-key add -
# apt update
# apt install anastasis-gtk
```

Do you have any questions?

https://anastasis.lu/

References:

1. Dennis Neufeld and Dominik Meister. *Anastasis: Password-less key recovery via multi-factormulti-party authentication*. **BFH, 2020**.
2. David Chaum, Christian Grothoff and Thomas Moser. *How to Issue a Central Bank Digital Currency*. **Swiss National Bank Working Papers, 3/2021**
3. Florian Dold. *The GNU Taler System: Practical and Provably Secure Electronic Payments*. **University of Rennes 1**, 2019.