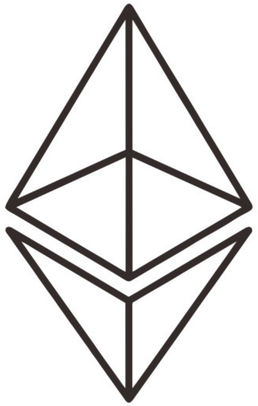


ERC1077 + 1078
Universal logins in a multi-device world

Or how to solve all our UX problems at the same time thanks to the magic of signed messages

Alex Van de Sande - Ethereum Foundation

Users…

- don’t care about ether
- don’t care about backing up private keys or seed phrases
- don’t understand why they can’t use standard 2fa
- don’t want huge hex-strings
- want a simple identifiable username
- don’t understand why they can’t use their credit card or appstore credit
- would rather not download anything on the desktop
- own multiple devices and switch between them constantly

Bad solution:

“Have a new account with ether on every device!”

Really bad solution:

“Login with our centralized proprietary system”

Terrible horrible very bad solution:

“Type your private key/seed phrase to login on this game!”

My humble suggestion

LET'S *NOT* TRAIN USERS TO GIVE AWAY THEIR KEYS

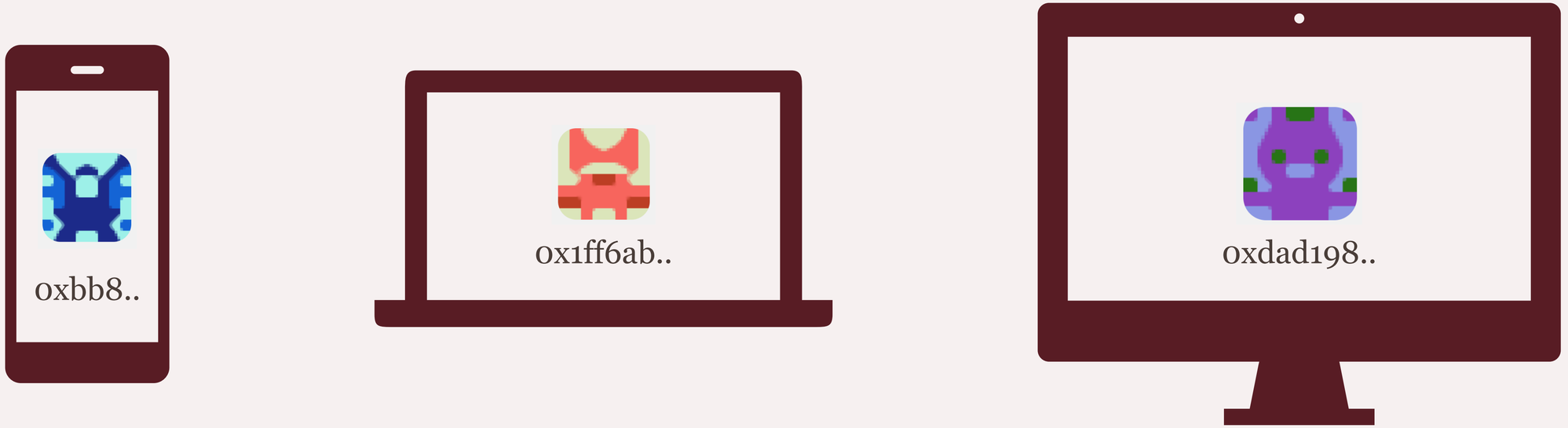
The crypto paradox

“This key is important. If you lose it, you lose all you money, therefore **make many backups**. But if it leaks, you lose all your money so **don’t have too many copies of it**.”

Solution

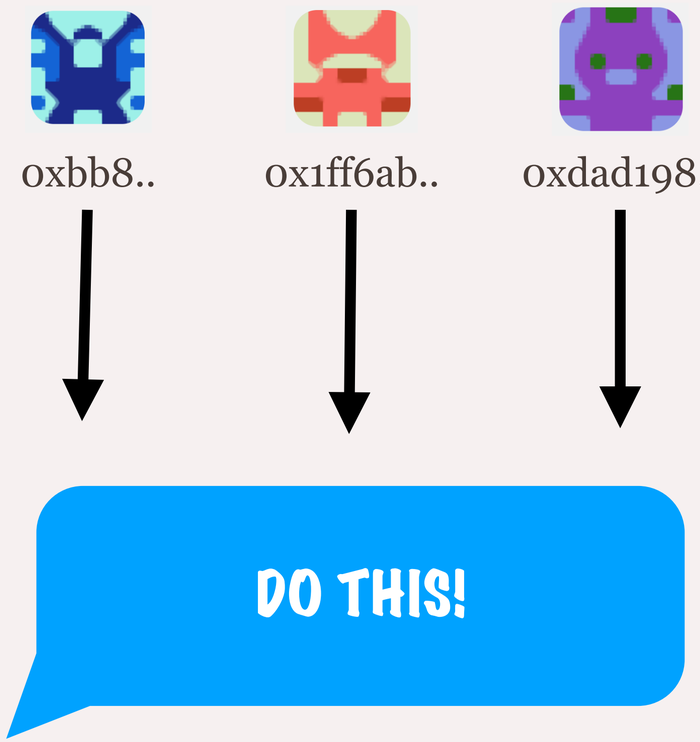
- Context specific ether less accounts
- Signed messages instead of direct transactions
- Gas relay abstraction on identity
- Identity contracts
- ENS Usernames
- Standard across multiple wallets

Context specific etherless accounts

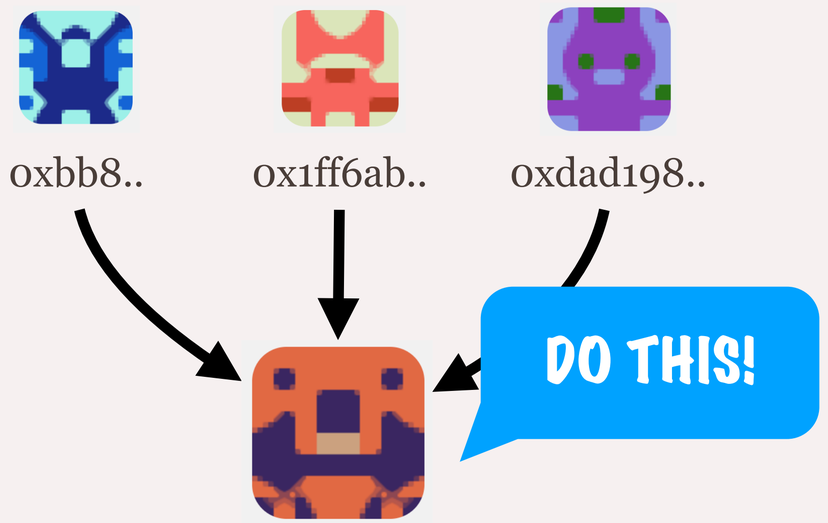


Generate private keys and protect them on the device. Don’t show it to user, don’t keep ether on it.

Signed messages



Identity Contract

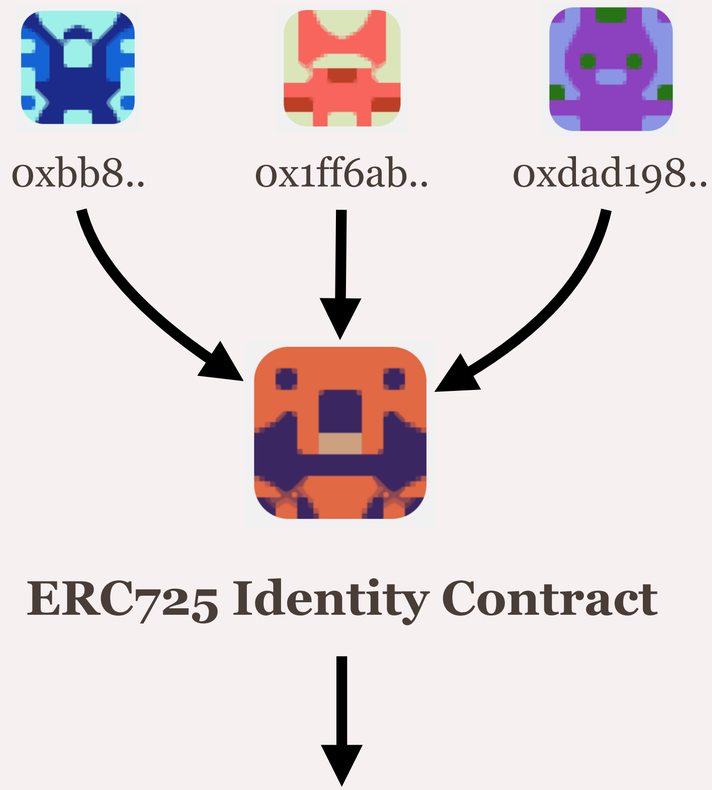


ERC725 Identity Contract

No ether, just used for signing

Are keys to an identity contract that holds ether and tokens

ENS Username

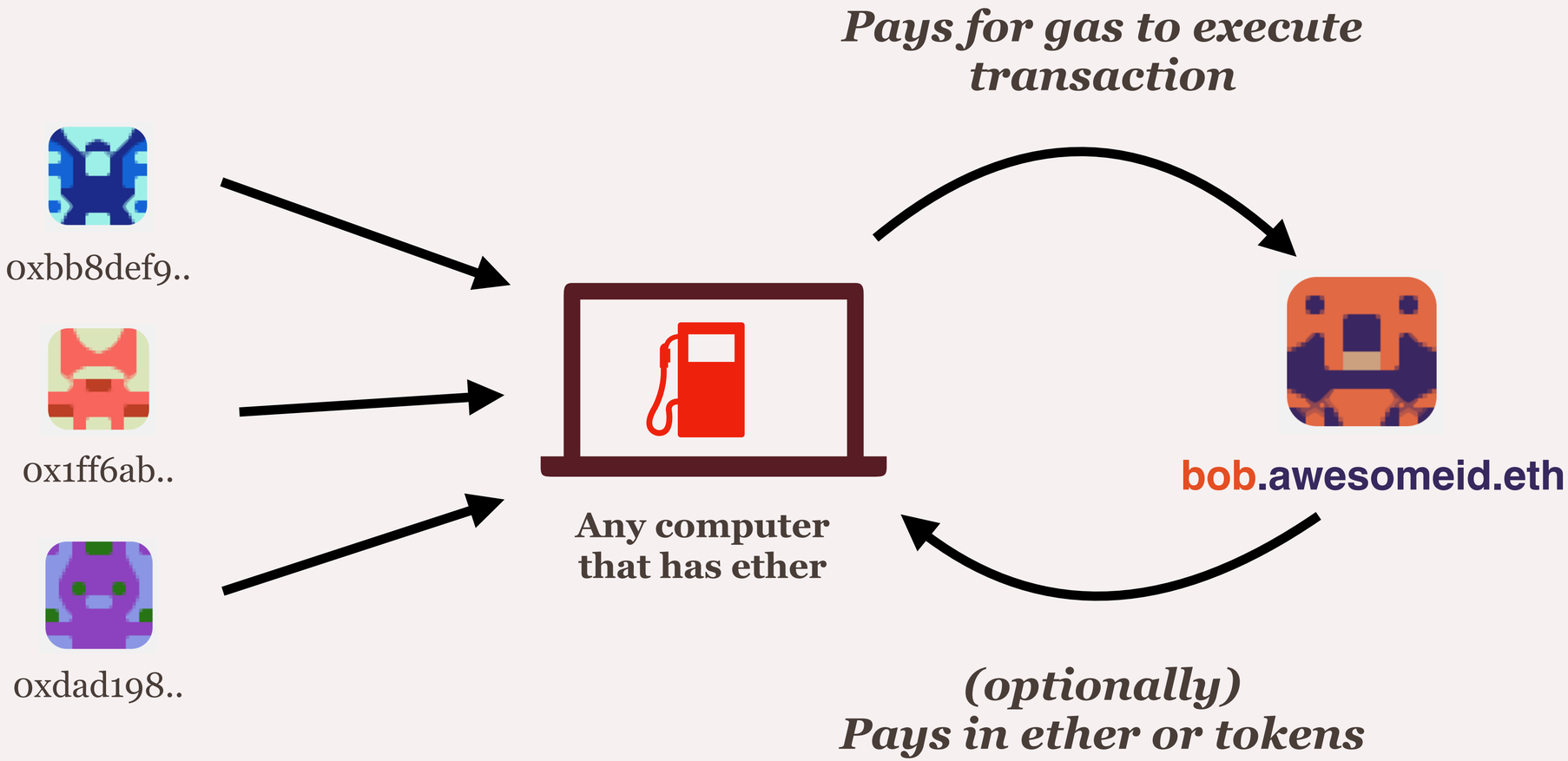


No ether, just used for signing

Are keys to an identity contract that holds ether and tokens

Register ENS domain/ subdomain to identity

Gas Relay Abstraction



ERC725 Identity

Hierarchy of keys:

0xbb8def9..

Encryption: lists any type of public key belonging to the user. *Ideal for communication, social media, games*

0x1ff6ab..

Action: can do actions in behalf of the identity
Can transfer ether, tokens, assets etc (part of multisig)

0xdad198..

Management: can do actions on the identity itself
Any sort of action in which to == self (include adding keys)

Recovery Options

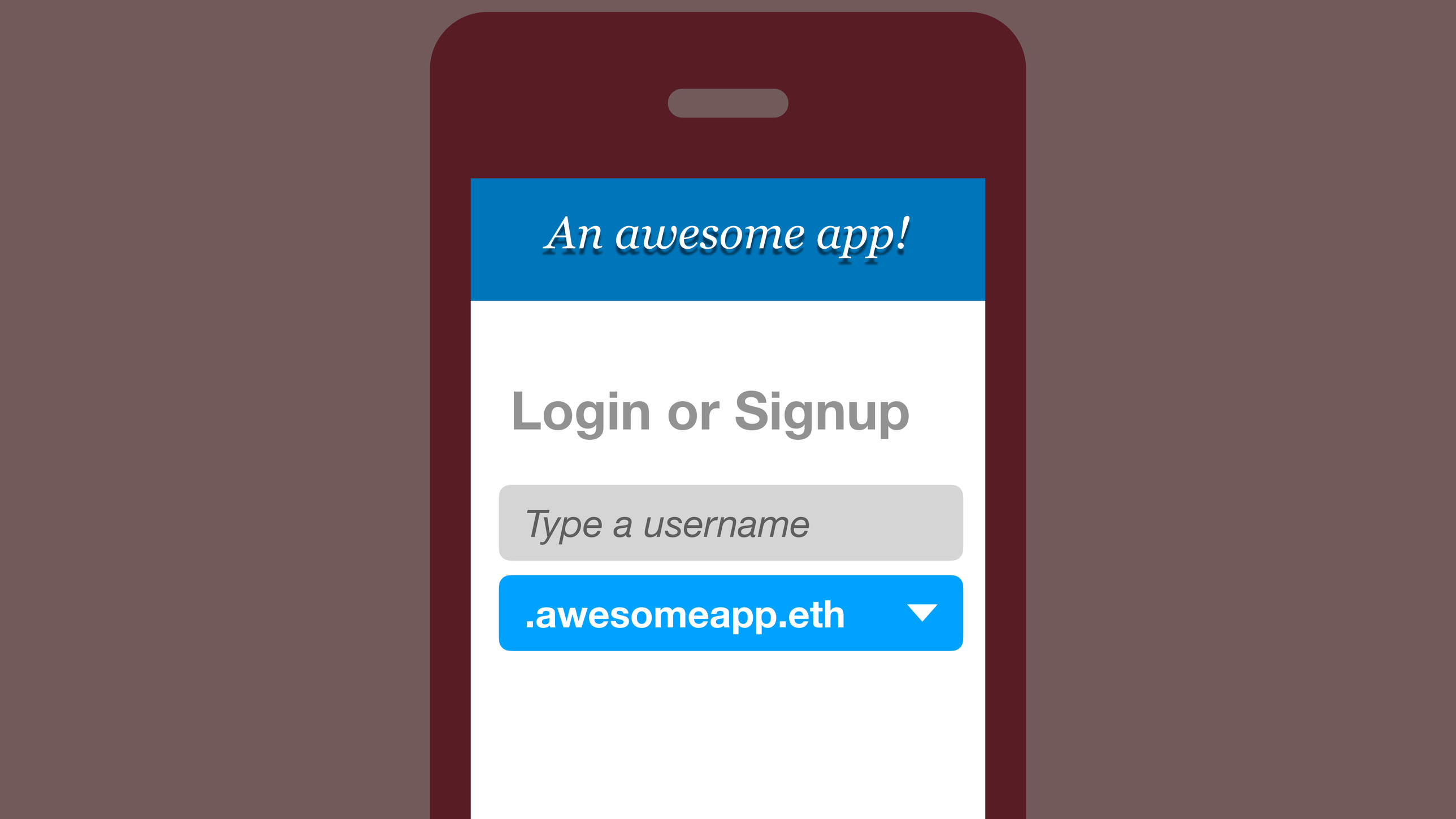
Possible fail safe measures

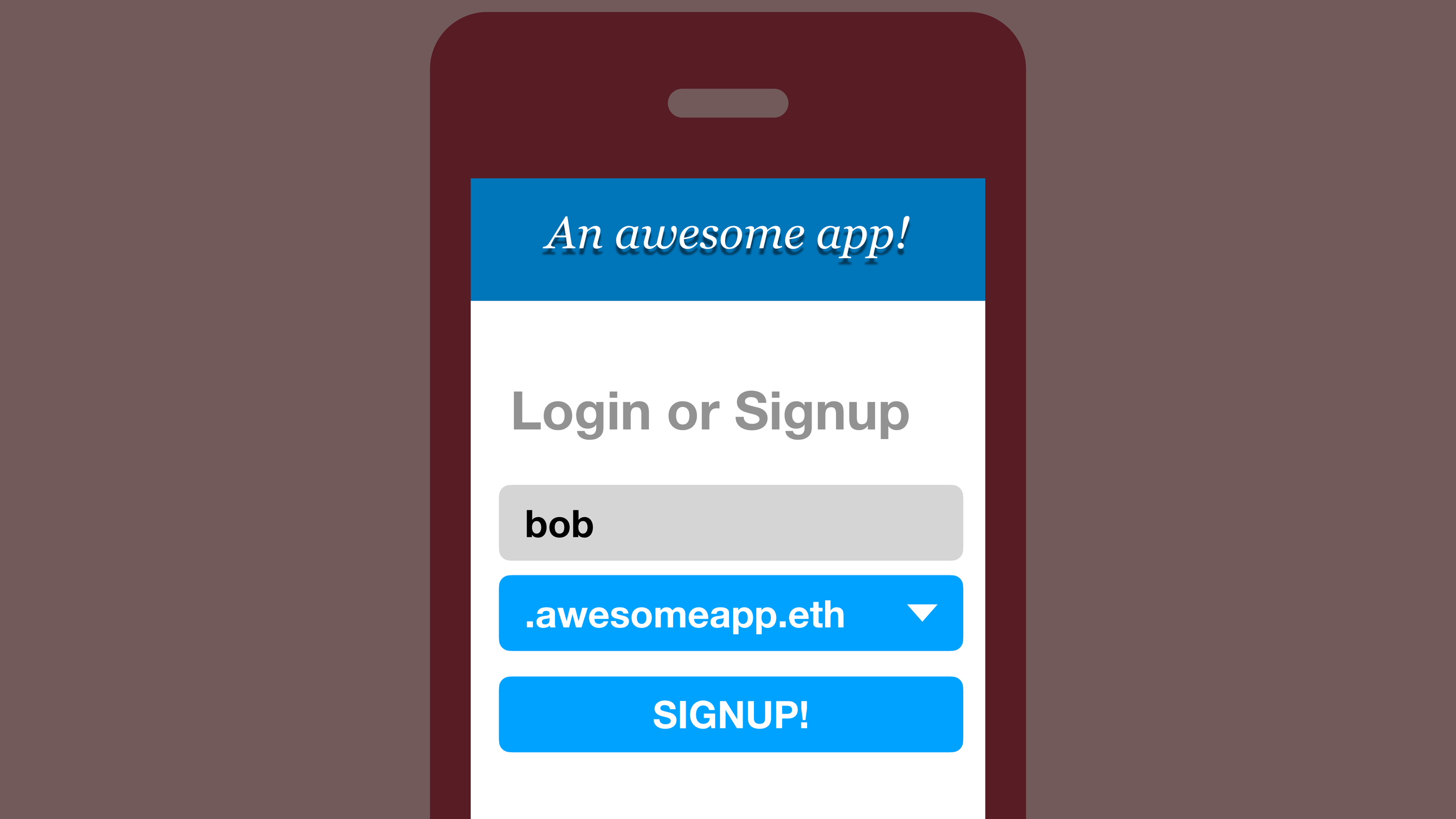
Friends Recover: encrypted list of “trusted friends” that can reset your identity *(being implemented by Status!)*

Central recovery: some services can allow recovery by email or manually checking documentation (which can disabled by user)

Dead man switch: allow specially appointed IDs to gain access to your ID if you fail to send any tx in a year

Can you repeat that, but with pics?


An awesome app!
Login or Signup
Type a username
.awesomeapp.eth


An awesome app!
Login or Signup
bob
.awesomeapp.eth
SIGNUP!

An awesome app!

Welcome, bob!

Cool Crypto App!

Cool kids are doing it!

Type a username

.cryptogame.eth

Cool Crypto App!

Cool kids are doing it!

Type a username

.cryptogame.eth

.mycrypto.eth

.aragon.eth

.eth

Cool Crypto App!

Cool kids are doing it!

bob.awesomeapp

.eth

CONNECT

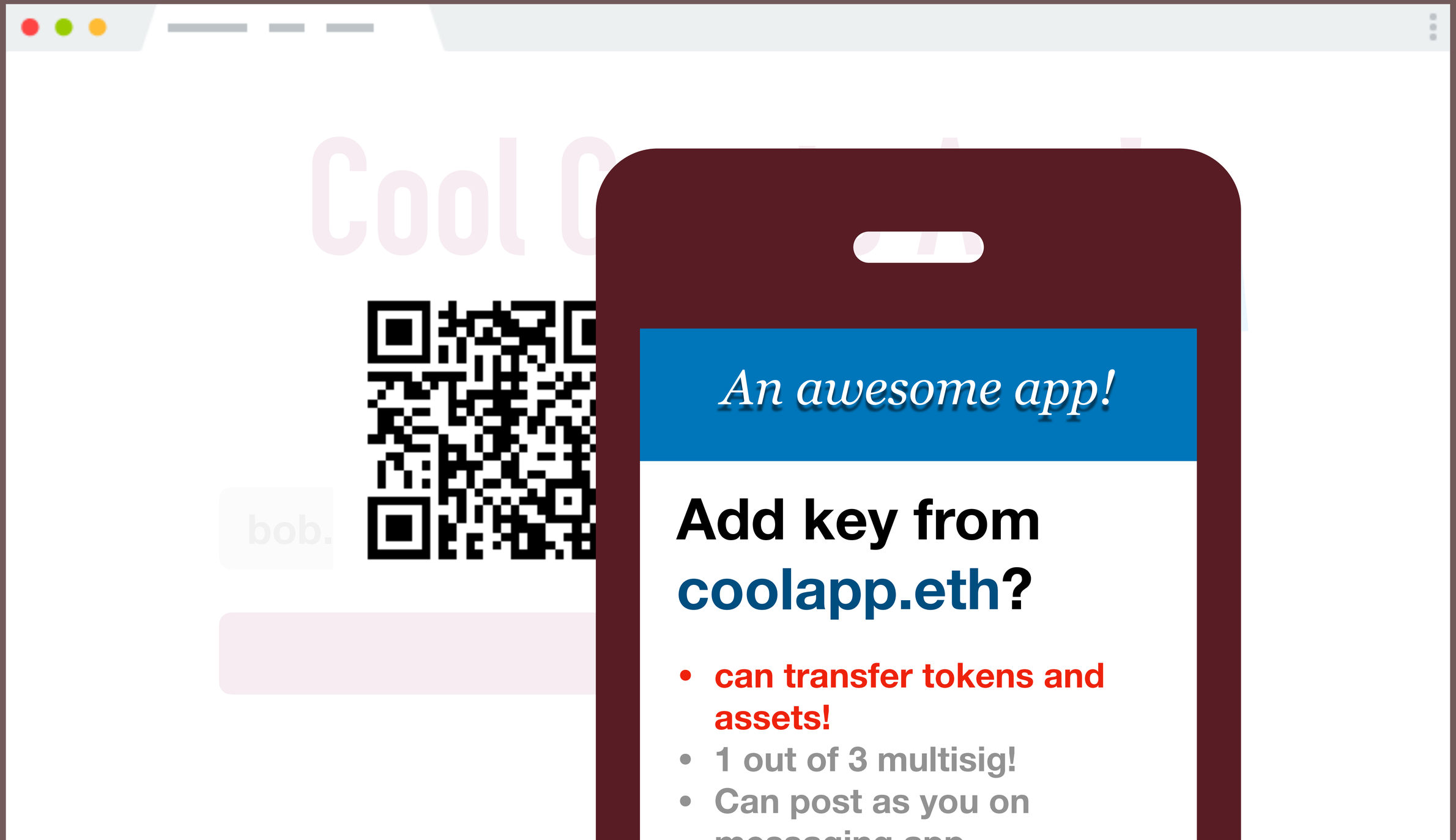
Cool Crypto App!

Cool kids are doing it!

SCAN THIS!

bob.

CONNECT


bob.
An awesome app!
Add key from coolapp.eth?
• can transfer tokens and assets!
• 1 out of 3 multisig!
• Can post as you on

Cool Crypto App!

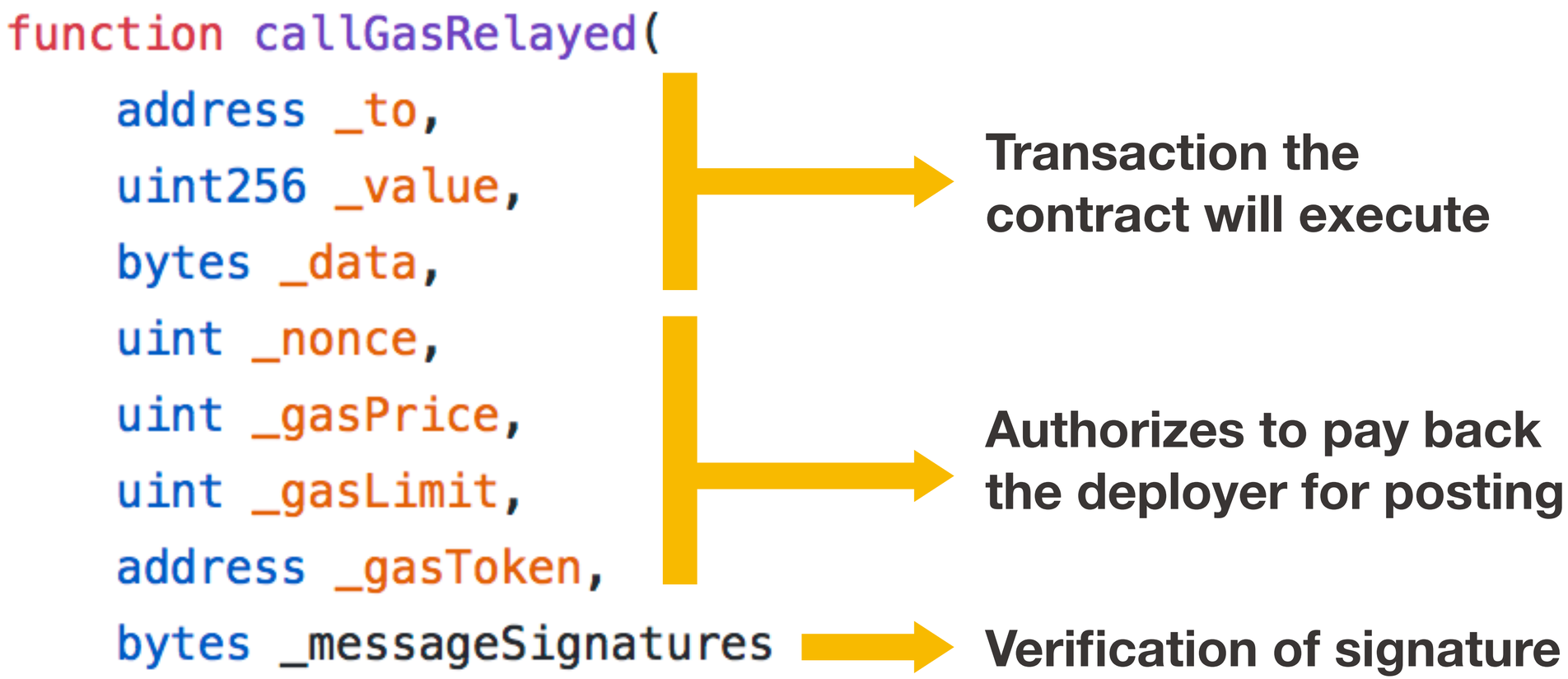
Cool kids are doing it!

Welcome, bob.awesomeapp.eth!

More security

- ✓ Every device or wallet you install works as a new security factor
- ✓ Apps can distribute usernames that can be used in other apps

Can you repeat that, but with code?


function callGasRelayed(
address _to,
uint256 _value,
bytes _data,
uint _nonce,
uint _gasPrice,
uint _gasLimit,
address _gasToken,
bytes _messageSignatures
)
Transaction the contract will execute
Authorizes to pay back the deployer for posting
Verification of signature

```
//verify if signatures are valid and came from correct actors;
verifySignatures(
    _to == address(this) ? MANAGEMENT_KEY : ACTION_KEY,
    signHash,
    _messageSignatures
);

//executes transaction
nonce++;
bool success = _to.call.value(_value)(_data);
emit ExecutedGasRelayed(
    signHash,
    success
);

//refund gas used using contract held ERC20 tokens or ETH
if (_gasPrice > 0) {
    uint256 _amount = 21000 + (startGas - gasleft());
    _amount = _amount * _gasPrice;
    if (_gasToken == address(0)) {
        address(msg.sender).transfer(_amount);
    } else {
        ERC20Token(_gasToken).transfer(msg.sender, _amount);
    }
}
```

Identity contract verifies signed message

Executes transaction from identity contract

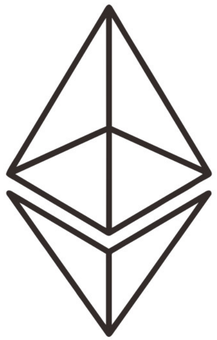
Pays back to the deployer in either token or ether

```
function verifySignatures(
    uint256 _requiredKey,
    bytes32 _signHash,
    bytes _messageSignatures
)
```

Allows multiple signatures to be posted only once

More flexibility

- ✓ Users do not need ether
- ✓ Apps can use any business model for their users (native platform credit, subscriptions, private tokens)
- ✓ Relaying as “desktop mining”



Help is welcome

eips.ethereum.org/eips/eip-1077

eips.ethereum.org/eips/eip-1078

Thanks to: Ricardo Schmidt from Status, Martin Köppelmann from Gnosis