Store now break it later?

JP Aumasson


~~Store now~~
break it later?
JP Aumasson
TAURUS

This talk

How to make blockchain data future-proof?

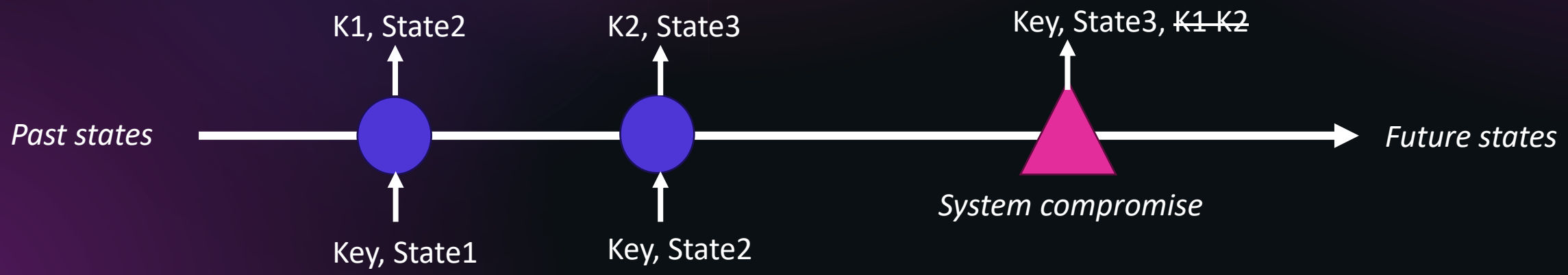
- From forward secrecy to perennial secrecy
- Examples of “secret data” and “bad events”
- What you should (not) do to be future-proof

Forward secrecy & Perennial secrecy

Forward secrecy

In key agreement protocols and PRNGs:

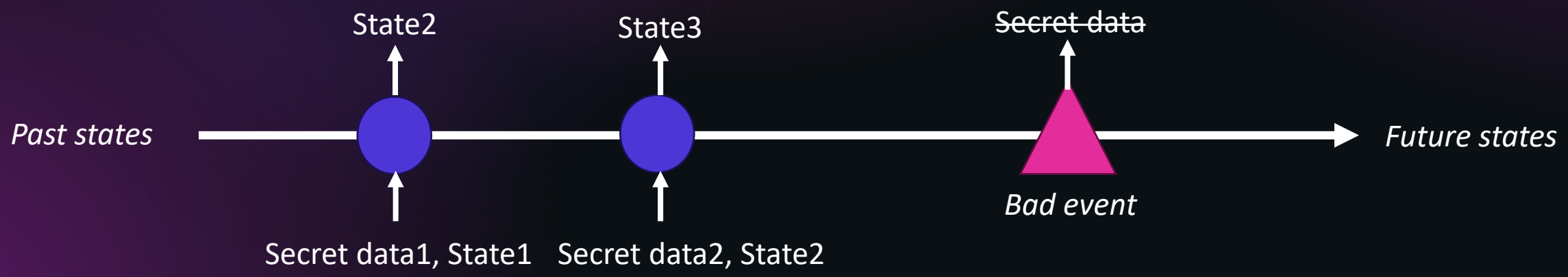
Guarantee that paste keys/randomness cannot be determined in the future from the system's state and keys



Perennial secrecy

Generalization of forward secrecy, suitable to blockchains:

The assurance that today's **secret data** will remain secret later even upon **bad events**



Example: PGP-encrypted email

IF **Secret data** = data encrypted using hybrid encryption

AND **Bad event** = private key compromised

AND Attacker obtains the ciphertext before or after the key is compromised

THEN NO Perennial Secrecy

Example: E2EE secure messaging

IF **Secret data** = Past Signal messages

AND **Bad event** = current keys compromised (long-term, ephemeral DH, and message keys)

THEN Perennial Secrecy

Because of the “symmetric key ratchet”

Example: E2EE secure messaging

IF **Secret data** = Past Signal messages

AND **Bad event** = device compromised

THEN It depends

Ephemeral messages that disappeared are safe

Messages still readable by the user are compromised

On-chain encrypted data

Unlike TLS or secure messaging, data is:

- Accessible, no need for eavesdropping capabilities
- Data stored forever and immutable (no “revocation”)

IF **Bad event** = Compromised keys

THEN NO Perennial Secrecy

Perennial secrecy

Generalization of forward secrecy:

In a system, the assurance that today's **secret data** will remain secret later even upon **bad events**

What **secret data** and what **bad events**?


Secret
data

On-chain data

Data directly **stored** (encrypted data, IPFS files, etc.)

Private **signing keys** (via their public keys)

ZK proofs (no encryption, but zero-knowledge'd data)

Private **signing keys** (via their public keys)

Fully-homomorphic encryption (FHE) ciphertexts

Aanti-MEV mempool **threshold encryption** ciphertexts

Off-chain data

Layer 2 data storage

Encrypted traffic between systems (such as nodes)

Credentials used to admin systems (passphrases, etc.)

API secrets, encrypted back-ups, wallet back-ups, etc.

On-chain commitments to secrets

Think “hash of a secret value” **Hash**(*secret*, *r*)

In theory: commitments offer the **hiding** property

Practical risks, esp. if randomness *r* is weak:

- Revealing low-entropy values
- Repudation/revocation impossible

Privacy-preserving systems

Privacy-preserving suggests that there is private data involved to be preserved (and thus which may fail to be so)

Private transfers or contracts/programs

Apps with ciphertexts, commitments, or hashes of:

- PII: name, geoloc, unique IDs, hardware IDs, IPs, etc.
- Biometric data (iris scans, fingerprints, etc.)

Bad events

Obvious bad events

Flawed keygen/entropy/PRNG (Profanity-like)

Smart contracts flaws

“Forgot to encrypt” (yes, this happens)

Known bad events

Broken scheme/protocol (ZKP, FHE, threshold schemes)

Key exposed or revoked, e.g. from a threshold quorum

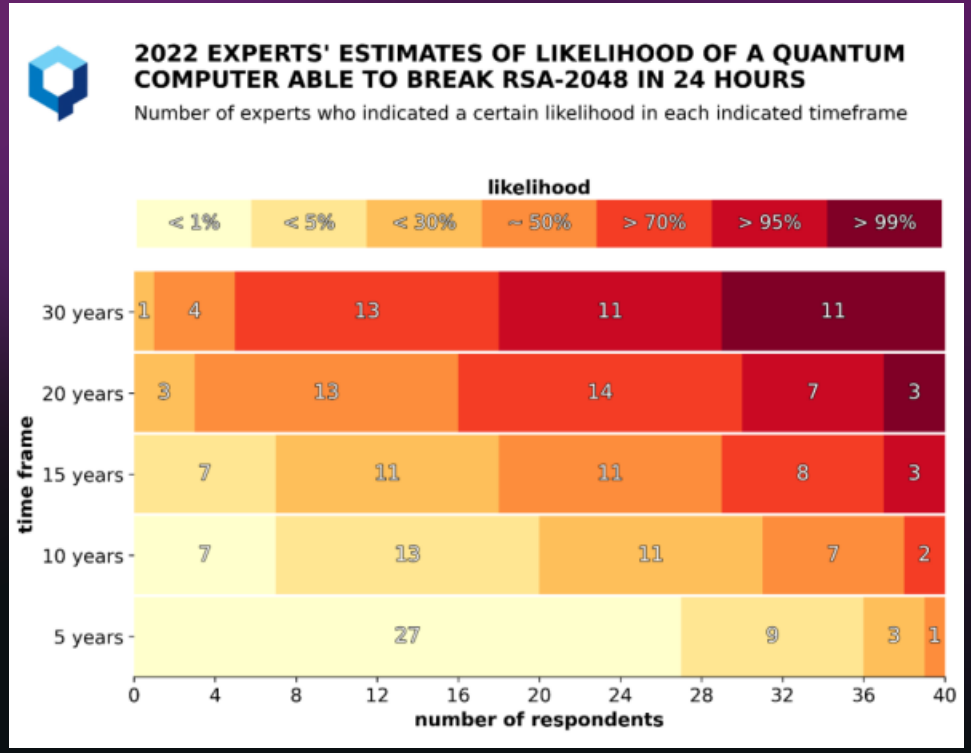
Quantum computers

Protocol side-channel attacks

Quantum computing

Signatures could “easily” recover via revocation and reissuance

Encrypted data compromised if public-key crypto used to encrypt or to generate keys



https://globalriskinstitute.org/publication/2022-quantum-threat-timeline-report/

Side channels

Remote Side-Channel Attacks on Anonymous Transactions

Florian Tramèr*
Stanford University

Dan Boneh
Stanford University

Kenneth G. Paterson
ETH Zürich

Abstract

Privacy-focused crypto-currencies, such as Zcash or Monero, aim to provide strong cryptographic guarantees for transaction confidentiality and unlinkability. In this paper, we describe side-channel attacks that let remote adversaries bypass these protections.

We present a general class of timing side-channel and traffic-analysis attacks on receiver privacy. These attacks enable an active remote adversary to identify the (secret) payee of any transaction in Zcash or Monero. The attacks violate the privacy goals of these crypto-currencies by exploiting side-channel information leaked by the implementation of different system components. Specifically, we show that a remote party can link all transactions that send funds to a user, by measuring the response time of that user's P2P node to certain requests. The timing differences are large enough that the attacks can be mounted remotely over a WAN. We responsibly disclosed the issues to the affected projects, and they have patched the vulnerabilities.

Can break FS is network traces recorded

ZK proofs leaks

The impact of a “break of ZKness” depends a lot on the application and proof system

ZKP values are often **succinct** (a few group elements):

may or may not be enough to leak sensitive data

Ex: 3 of ~200 bits in the Groth16 SNARK

Trivial bound log2(field size)

Key management failure

The hardest problem when actually using cryptography

Key management **will** fail

Murphy's laws of cryptography

- Cryptography turns a security problem into a key management problem.
- New cryptography generates new attacks.
- If it's provably secure, it's probably not.
- Any large enough system will include broken cryptography.
- Any attempt to standardize will instead lead to massive fragmentation.
- Any new standard is obsolete.
- Broken in theory does not imply broken in practice, and vice-versa.
- There's always a trusted third-party.
- What sounds like a solution now will create more problems later:
  - "Just use an HSM."
  - "Assume a PKI is available."
  - "Assume a broadcast channel."
  - "Assume little-endianness."
- Come for the cryptography, stay for the DER and PEM encodings.
- Any new cryptography API will use different conventions than all existing cryptography APIs.

Key rotation for risk reduction

Key rotation and “crypto periods” common in security infrastructure to reduce the risk of key compromise

On-chain encrypted data: can’t rotate keys

Threshold signing: can refresh shares, but not keys

Wallets: can “rotate” wallets (rebalancing)

Key leak from supply-chain flaws

An encryption (symmetric) key may come from a…

- Weak/compromised key generator
- Non-PQ handshake
- A KMS, 1000 ways it could go wrong
  (weak config, malicious owners, compromised cloud, etc.)

Hacks

People will fail: get phished, steal funds, get coerced, etc.

Systems will get hacked / broken , keys will leak

How to reduce the risk and damage?

Distribute trust, with collective key control

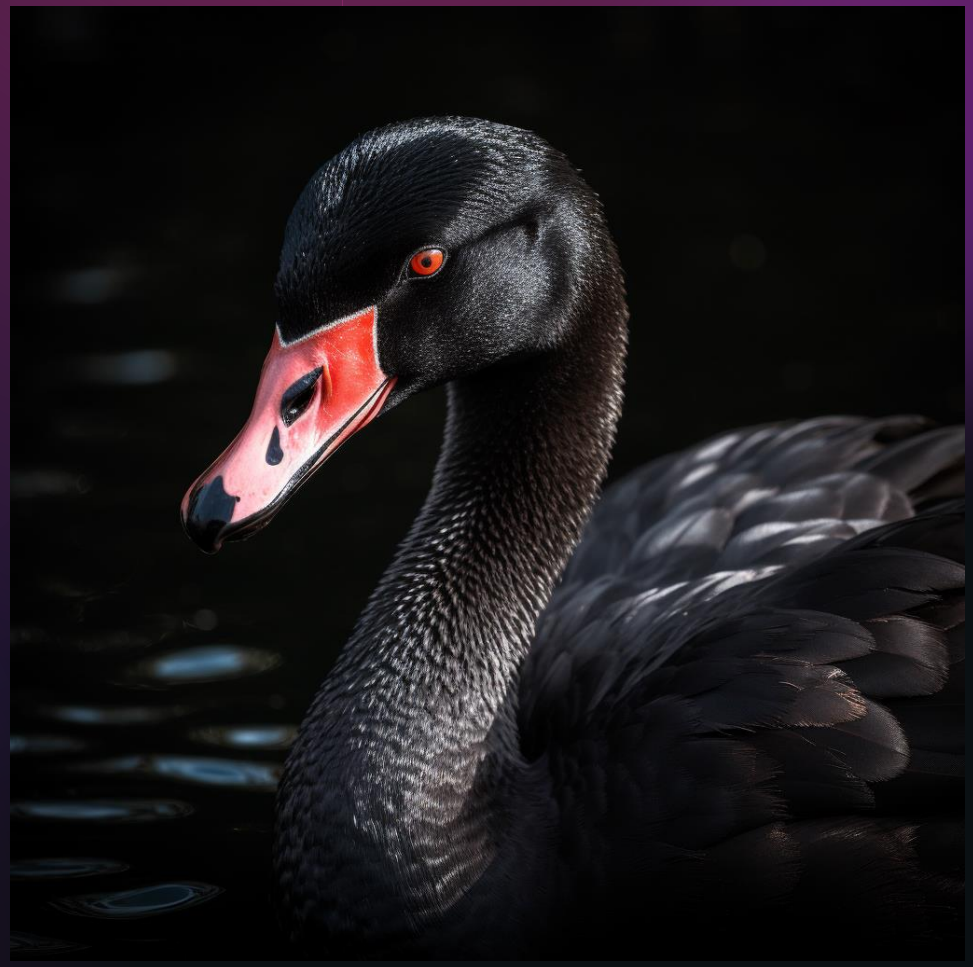
Reduce the blast radius (distribute assets, no master key)

Hacks

Monero Community Wallet Loses Entire Balance in Hack

The community crowdfunding wallet lost 2,675 XMR worth close to $460,000 in an attack, while the source of the vulnerability is still yet to be identified.

Unexpected bad events



What (not) to do?

Secret data

Don’t put it on chain! Even **encrypted**

If you have to, consider:

- Using randomized commitments instead of data
- Storing a pointer to off-chain (encrypted) data
- Storing encrypted keys on-chain and data off-chain

What data?

Avoid at all costs storing PII on-chain, even if encrypted

If such data needed, store a reference to off-chain storage

Data with short lifetime or secrecy requirement is by definition future-proof, so can be OK to log on-chain

Bad events

Identify them, try to reduced their likelihood

Do threat modelling:

- Identify of secret data (private inputs, etc.)
- Estimate their lifetime

Challenges

- Perennial security of systems deployed today is rarely evaluated prior to deployment
- Blockchain projects often have low security, due to the low maturity: minimal processes and accountability