Pixel: Multi-Signatures for Consensus

Manu Drijvers (DFINITY)
Sergey Gorbunov (Algorand & University of Waterloo)
Gregory Neven (DFINITY)
Hoeteck Wee (Algorand & CNRS, ENS, PSL)
Permissioned/Proof-of-Stake Blockchains

**Consensus**: nodes agree on sequence of blocks

**Proof of stake (PoS)**: nodes vote on block proposals, weighted by stake  
  e.g., Algorand, Cardano, Ethereum Casper

**Permissioned**: nodes vote by access structure  
  e.g., Ripple, Hyperledger Fabric
Permissioned/Proof-of-Stake Blockchains

**Proof of stake (PoS):** nodes **sign** block proposals, weighted by stake
e.g., Algorand, Cardano, Ethereum Casper

**Permissioned:** nodes **sign** by access structure
e.g., Ripple, Hyperledger Fabric
Multi-Signatures in Blockchains

- Single multi-signature $\Sigma$ under $pk_1,\ldots,pk_n$ on $m$
- Short signature, efficient verification
  - (preferably $\approx$ single signature)

[IN83, OO91, MOR01, BLS01, B03, BN06, BDN18, ...]
The Problem of Posterior Corruption \cite{BPS16} 
aka long-range attacks \cite{B15}, costless simulation \cite{P15}

Chain integrity assumption:
\leq \text{fraction } f \text{ of nodes/stake corrupt}
The Problem of Posterior Corruption [BPS16]

aka long-range attacks [B15], costless simulation [P15]

Chain integrity assumption:

\[ \leq \text{fraction } f \text{ of nodes/stake corrupt} \]
The Problem of Posterior Corruption [BPS16]
aka long-range attacks [B15], costless simulation [P15]

Chain integrity assumption:
\[ \leq \text{fraction } f \text{ of nodes/stake signing keys corrupt} \]
The Problem of Posterior Corruption \cite{BPS16}
aka long-range attacks \cite{B15}, costless simulation \cite{P15}

Chain integrity assumption:
\leq \text{fraction } f \text{ of node/stake keys corrupt}
The Problem of Posterior Corruption [BPS16]

aka long-range attacks [B15], costless simulation [P15]

Chain integrity assumption:
\[ \leq \text{fraction } f \text{ of node/stake keys corrupt at any point in the past!} \]

Long after nodes left, sold stake

Aggravated by committee signing (adaptive attacks)
Solution: Forward-Secure Signatures

\[(pk, sk_0) \leftarrow \text{KeyGen}, \ sk_{t+1} \leftarrow \text{Update}(sk_t)\]

\[\sigma \leftarrow \text{Sign}(sk_t, m), \ b \leftarrow \text{Verify}(pk, t, m)\]

\[sk_t\] can’t be used to forge signatures \(t\)
Pixel: Forward-Secure Multi-Signatures

Bilinear map $e: G_1 \times G_2 \rightarrow G_T$, generators $g_1, g_2$ of $G_1, G_2$

$pk = g_2^x$ for $x \xleftarrow{\$} \mathbb{Z}_q$

$sk_t$: HIBE-style binary tree [CHK03, BBG05]
**Pixel : Forward-Secure Multi-Signatures**

**Bilinear map** \( e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T \), generators \( g_1, g_2 \) of \( \mathbb{G}_1, \mathbb{G}_2 \)

**pk** = \( g_2^x \) for \( x \leftarrow \mathbb{R} \) \( \mathbb{Z}_q \)

**sk_t** : HIBE-style binary tree [CHK03, BBG05]

**Public parameters** \( h, h_0, ..., h_\ell \in \mathbb{G}_1 \)

\[
H(t,m) = h_0 \cdot \prod h_i^{t_i} \cdot h_\ell^{H'(m)} \text{ where } t = t_1...t_{\ell-1}
\]

**Sign**(sk_t, m) : ( \( h^x \cdot H(t,m)^r \), \( g_2^r \))

**Verify**(pk, t, m) : \( e( h, pk ) \cdot e( H(t,m) ) \)

**Aggregate** : ( \( \prod_{i=1}^N h_i^{x_i} \cdot H(t,m)^{r_i} \), \( \prod_{i=1}^N g_2^{r_i} \) ), verify wrt \( \text{apk} = \prod_{i=1}^N pk_i \)
Pixel: Features

- **Provably secure** in random-oracle model assuming secure erasures and hardness of $\log(T)$ weak bilinear Diffie-Hellman inversion (wBDHI*)
- Performance on BLS12-381 curve with 1500 signers, $2^{32}$ time periods
  - **small:** pk 48 B, multi-signature 144 B
  - **fast:** sign 2.8 ms (4 exp), aggregate 7.2 ms (N mult), verify aggregate 6.7 ms (3 pair + 1 exp), key update 1.8 ms (2 exp)
  - trade off key/signature sizes vs computation by switching $\mathbb{G}_1$ and $\mathbb{G}_2$
- No trusted setup (hash to curve)
- Rogue key protection via proofs of possession [RY07]
Integration into Algorand blockchain

1500 certifying votes (signatures) per block
≈ 33% savings in blockchain size and block verification time compared to BM-Ed25519