An isogeny-based adaptor signature using SQISign

Valerie Gilchrist, David Jao

University of Waterloo

March 9, 2023

Payment Channel Networks

Blockchain transactions can be very costly.

Payment Channel Networks

Blockchain transactions can be very costly.

Payment Channel Networks

Blockchain transactions can be very costly.

Payment Channel Networks

Blockchain transactions can be very costly.

Payment Channel Networks

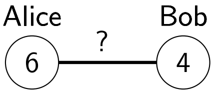
Blockchain transactions can be very costly.

Payment Channel Networks

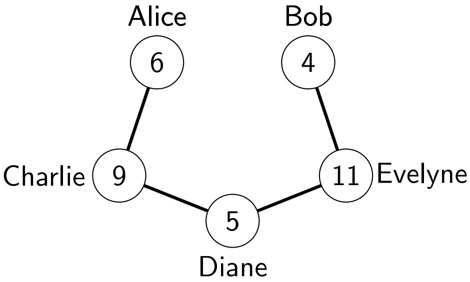
Blockchain transactions can be very costly.

Payment Channel Networks

Blockchain transactions can be very costly.

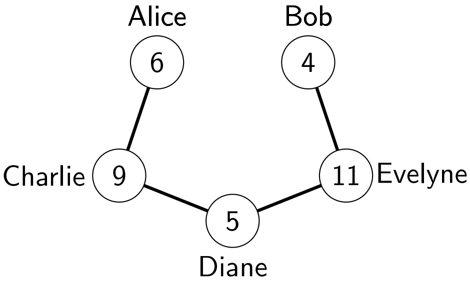


Blockchain transactions can be very costly.



Payment Channel Networks

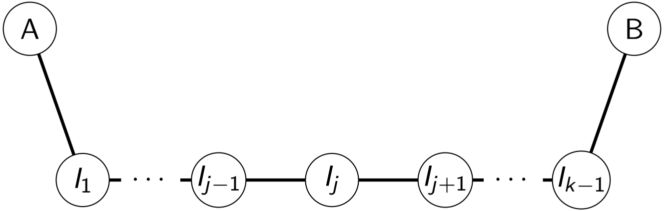
Blockchain transactions can be very costly.



How can Alice be assured her money will arrive to Bob?

Anonymous Multi-Hop Locks (AMHL)

Anonymous Multi-Hop Locks (AMHL)



Anonymous Multi-Hop Locks (AMHL)

Set-Up:

Anonymous Multi-Hop Locks (AMHL)

Set-Up:

Alice first chooses a cryptographic hard problem

$$f : \mathcal{L}_{witness} \rightarrow \mathcal{L}_{statement}$$

e.g. (x, g^x) is a witness, statement pair for the discrete logarithm problem

Anonymous Multi-Hop Locks (AMHL)

Set-Up:

Alice first chooses a cryptographic hard problem

$$f : \mathcal{L}_{witness} \rightarrow \mathcal{L}_{statement}$$

e.g. (x, g^x) is a witness, statement pair for the discrete logarithm problem
Next, she will choose a random collection of elements

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathcal{L}_{witness}.$$

Anonymous Multi-Hop Locks (AMHL)

Set-Up:
Alice first chooses a cryptographic hard problem

$$f : \mathcal{L}_{witness} \to \mathcal{L}_{statement}$$

e.g. (x, g^x) is a witness, statement pair for the discrete logarithm problem
Next, she will choose a random collection of elements

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathcal{L}_{witness}.$$

She will then compute the following for each $j \in [1, \cdots k-1]$:

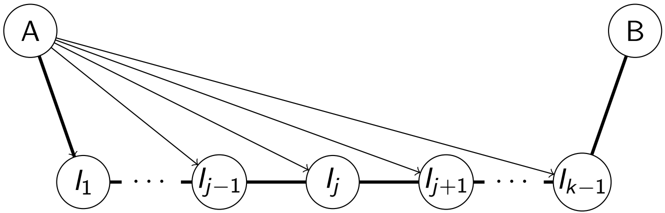
$$y_j = \sum_{i=0}^{j} \ell_i, \, Y_j = f(y_j)$$

Anonymous Multi-Hop Locks (AMHL)

Commit:

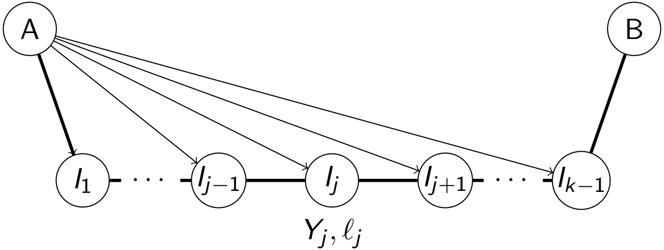
Anonymous Multi-Hop Locks (AMHL)

Commit:



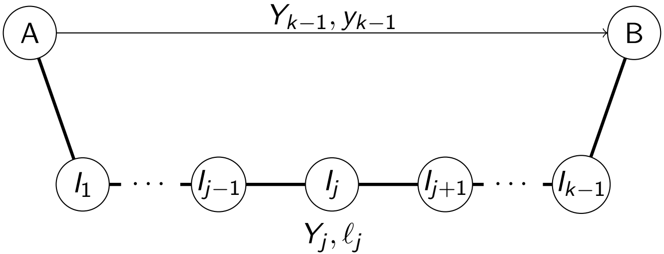
Anonymous Multi-Hop Locks (AMHL)

Commit:



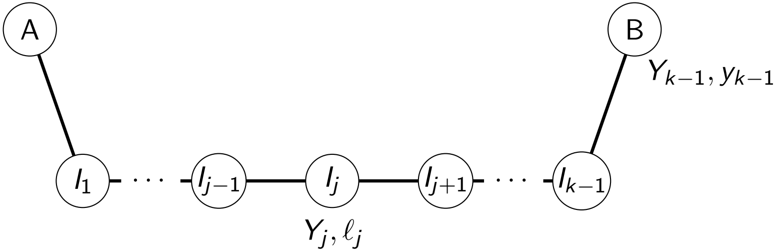
Anonymous Multi-Hop Locks (AMHL)

Commit:



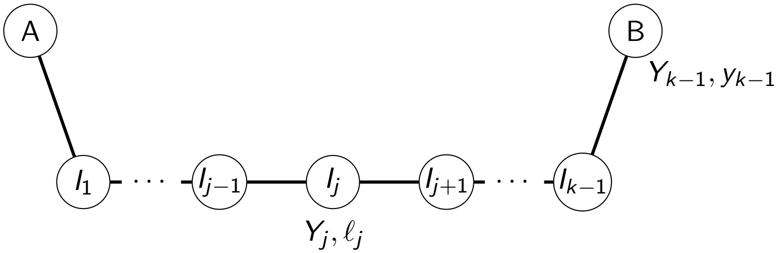
Anonymous Multi-Hop Locks (AMHL)

Commit:



Anonymous Multi-Hop Locks (AMHL)

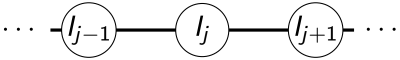
Commit:



Intermediary I_j will sign a contract agreeing to release funds to I_{j+1} on the condition that I_{j+1} can provide y_j.

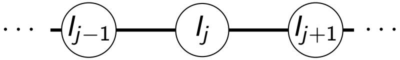
Anonymous Multi-Hop Locks (AMHL)

Release:



Anonymous Multi-Hop Locks (AMHL)

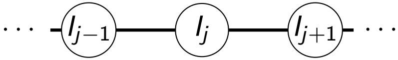
Release:



$$y_j \leftarrow l_{j+1}$$

Anonymous Multi-Hop Locks (AMHL)

Release:

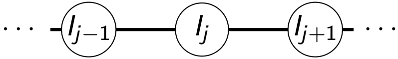


$$y_j \leftarrow l_{j+1}$$

$$y_{j-1} = y_j - \ell_j$$

Anonymous Multi-Hop Locks (AMHL)

Release:



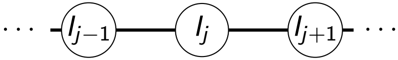
$$y_j \leftarrow l_{j+1}$$

$$y_{j-1} = y_j - \ell_j$$

$$l_j \leftarrow y_{j-1}$$

Anonymous Multi-Hop Locks (AMHL)

Release:



$$y_j \leftarrow l_{j+1}$$

$$y_{j-1} = y_j - \ell_j$$

$$l_j \leftarrow y_{j-1}$$

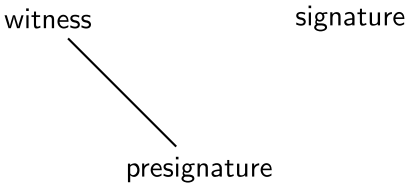
...how can we make this post-quantum?

Adaptor Signatures

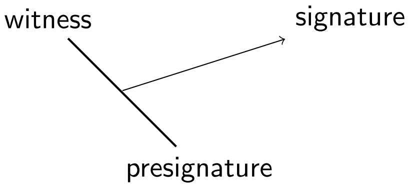
witness signature

presignature

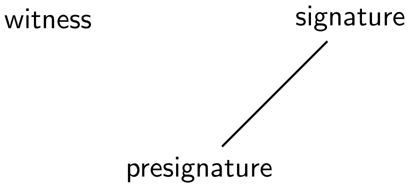
Adaptor Signatures



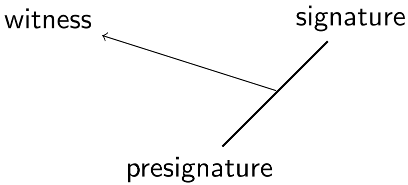
Adaptor Signatures



Adaptor Signatures



Adaptor Signatures



Adaptor Signatures

Let R be a hard relation, and $(y, Y) \in \mathrm{R}$.

Adaptor Signatures

Let R be a hard relation, and $(y, Y) \in$ R.
Consider a signature scheme, Σ, consisting of three algorithms:

Adaptor Signatures

Let R be a hard relation, and $(y, Y) \in \mathsf{R}$.
Consider a signature scheme, Σ, consisting of three algorithms:

$$\mathsf{KeyGen}(\lambda) \rightarrow \mathsf{sk}, \mathsf{pk}$$
$$\mathsf{Sig}(\mathsf{sk}, m) \rightarrow \sigma$$
$$\mathsf{Ver}(\mathsf{pk}, m, \sigma) \rightarrow b \in \{0, 1\}$$

Adaptor Signatures

Let R be a hard relation, and $(y, Y) \in \mathrm{R}$.
Consider a signature scheme, Σ, consisting of three algorithms:

$$\mathsf{KeyGen}(\lambda) \to \mathsf{sk}, \mathsf{pk}$$
$$\mathsf{Sig}(\mathsf{sk}, m) \to \sigma$$
$$\mathsf{Ver}(\mathsf{pk}, m, \sigma) \to b \in \{0, 1\}$$

Then an adaptor signature scheme with respect to R and Σ consists of four algorithms:

$$\mathsf{PreSig}(\mathsf{sk}, m, Y) \to \widetilde{\sigma}$$
$$\mathsf{PreVer}(\mathsf{pk}, m, Y, \widetilde{\sigma}) \to b \in \{0, 1\}$$
$$\mathsf{Adapt}(\widetilde{\sigma}, y) \to \sigma$$
$$\mathsf{Extract}(\sigma, \widetilde{\sigma}, Y) \to y$$

Example

Schnorr Signature

Alice chooses a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Example

Schnorr Signature

Alice chooses a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.

She publishes $X = g^x$ as her public key.

Example

Schnorr Signature

Alice chooses a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.

She publishes $X = g^x$ as her public key.

For a message $m \in \{0,1\}^*$, she chooses $k \in \mathbb{Z}_q$ and computes $r := \mathcal{H}(X || g^k || m)$ and $s := k + rx$.

Alice's signature is $\sigma = (r, s)$.

Example

Schnorr Signature

Alice chooses a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.

She publishes $X = g^x$ as her public key.

For a message $m \in \{0,1\}^*$, she chooses $k \in \mathbb{Z}_q$ and computes $r := \mathcal{H}(X || g^k || m)$ and $s := k + rx$.

Alice's signature is $\sigma = (r, s)$.

A verifier will check that $r = \mathcal{H}(X || g^s X^{-r} || m)$.

Example

Schnorr-based Adaptor Signature

She chooses $\mathsf{R}_g = \{(y, Y) | Y = g^y\} \subseteq \mathbb{G} \times \mathbb{Z}_q$.

Example

Schnorr-based Adaptor Signature

She chooses $\mathrm{R}_g = \{(y, Y) | Y = g^y\} \subseteq \mathbb{G} \times \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.
She publishes $X = g^x$ as her public key.

Example

Schnorr-based Adaptor Signature

She chooses $\mathrm{R}_g = \{(y, Y) | Y = g^y\} \subseteq \mathbb{G} \times \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.
She publishes $X = g^x$ as her public key.

For a message $m \in \{0,1\}^*$, she chooses $k \in \mathbb{Z}_q$ and computes $r := \mathcal{H}(X || g^k Y || m)$ and $s := k + rx$.

Example

Schnorr-based Adaptor Signature

She chooses $\mathrm{R}_g = \{(y, Y) | Y = g^y\} \subseteq \mathbb{G} \times \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.
She publishes $X = g^x$ as her public key.

For a message $m \in \{0, 1\}^*$, she chooses $k \in \mathbb{Z}_q$ and computes $r := \mathcal{H}(X || g^k Y || m)$ and $s := k + rx$.

Alice's presignature is $\widetilde{\sigma} = (r, s)$.
Her signature is $s' = s + y$.

Example

Schnorr-based Adaptor Signature

She chooses $\mathrm{R}_g = \{(y, Y) | Y = g^y\} \subseteq \mathbb{G} \times \mathbb{Z}_q$.

Alice chooses her secret key $x \in \mathbb{Z}_q$.
She publishes $X = g^x$ as her public key.

For a message $m \in \{0,1\}^*$, she chooses $k \in \mathbb{Z}_q$ and computes $r := \mathcal{H}(X || g^k Y || m)$ and $s := k + rx$.

Alice's presignature is $\widetilde{\sigma} = (r, s)$.
Her signature is $s' = s + y$.

A verifier will check that $r = \mathcal{H}(X || g^{s'} X^{-r} || m)$.

AMHL via Adaptor Sigantures

Setup:

AMHL via Adaptor Sigantures

Setup:

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathcal{L}_{witness}.$$

For each $j \in [1, \cdots k-1]$:

$$y_j = \sum_{i=0}^{j} \ell_i, \, Y_j = f(y_j)$$

AMHL via Adaptor Sigantures

Setup:

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathcal{L}_{witness}.$$

For each $j \in [1, \cdots k-1]$:

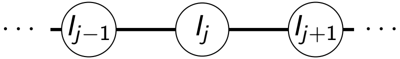
$$y_j = \sum_{i=0}^{j} \ell_i,\, Y_j = f(y_j)$$

Commit:

Each I_j will create a pre-signature $\hat{\sigma}_i = \mathrm{PreSig}(\mathrm{sk_i}, \mathrm{tx_i}, \mathrm{Y_i})$ where $\mathrm{tx_i}$ is the conditional contract stating that I_j will release funds to I_{j+1} once I_j is provided their full signature.

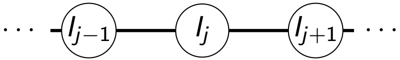
AMHL via Adaptor Sigantures

Release:



AMHL via Adaptor Sigantures

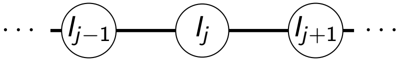
Release:



$$\sigma_j \leftarrow I_{j+1}$$

AMHL via Adaptor Sigantures

Release:

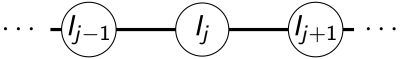


$$\sigma_j \leftarrow I_{j+1}$$

$$y_j \leftarrow \mathsf{Extract}(\sigma_j, \widetilde{\sigma}_j, Y_j)$$

AMHL via Adaptor Sigantures

Release:



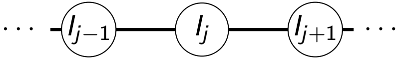
$$\sigma_j \leftarrow I_{j+1}$$

$$y_j \leftarrow \text{Extract}(\sigma_j, \widetilde{\sigma}_j, Y_j)$$

$$y_{j-1} = y_j - \ell_j$$

AMHL via Adaptor Sigantures

Release:



$$\sigma_j \leftarrow I_{j+1}$$

$$y_j \leftarrow \mathsf{Extract}(\sigma_j, \widetilde{\sigma}_j, Y_j)$$

$$y_{j-1} = y_j - \ell_j$$

$$\sigma_{j-1} \leftarrow \mathsf{Adapt}(\widetilde{\sigma}_{j-1}, y_{j-1})$$

Post-quantum adaptor signatures

Currently there are two post-quantum adaptor signature schemes:

- Lattice Adaptor Signature (LAS) using Dilithium (Esgin, Ersoy, Erkin, 2020).
- Isogeny Adaptor Signature (IAS) using CSI-FiSh (Tairi, Moreno-Sanchez, Maffei, 2021).
  - Derived from CSIDH.
  - May not be secure for some instances.

Post-quantum adaptor signatures

Currently there are two post-quantum adaptor signature schemes:

- Lattice Adaptor Signature (LAS) using Dilithium (Esgin, Ersoy, Erkin, 2020).
- Isogeny Adaptor Signature (IAS) using CSI-FiSh (Tairi, Moreno-Sanchez, Maffei, 2021).
  - Derived from CSIDH.
  - May not be secure for some instances.

A generic construction was published (Erwig, Faust, Hostakova, Maitra, Riahi, 2021), but does not include most post-quantum signatures, such as SQISign.

Post-quantum adaptor signatures

Currently there are two post-quantum adaptor signature schemes:

- Lattice Adaptor Signature (LAS) using Dilithium (Esgin, Ersoy, Erkin, 2020).
- Isogeny Adaptor Signature (IAS) using CSI-FiSh (Tairi, Moreno-Sanchez, Maffei, 2021).
  - Derived from CSIDH.
  - May not be secure for some instances.

A generic construction was published (Erwig, Faust, Hostakova, Maitra, Riahi, 2021), but does not include most post-quantum signatures, such as SQISign.

The latest generic construction (Dai, Okamoto, Yamamoto,2022) covers all signature schemes, but Extract and Adapt are trivial.

SQISign

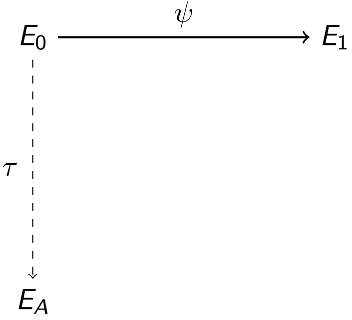
SQISign

E_0

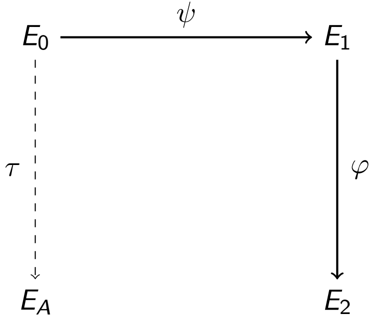
SQISign

$$E_0 \xrightarrow{\ \tau\ } E_A$$

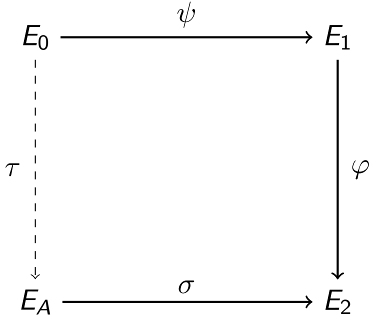
SQISign



SQISign



SQISign



SQISign Adaptor Signature (SAS)

Let (P_0, Q_0) be a basis for $E_0[\ell^e]$, for some small prime ℓ.
We choose our hard relation to be

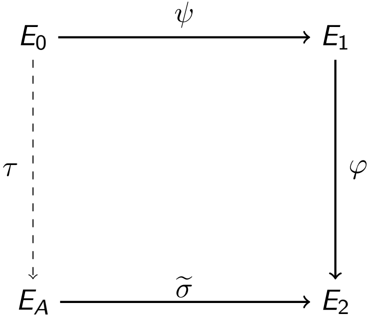
$$\mathrm{R}_{SSI} := \{(y, E_Y) | y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle\}$$

SQISign Adaptor Signature (SAS)

Let (P_0, Q_0) be a basis for $E_0[\ell^e]$, for some small prime ℓ.
We choose our hard relation to be

$$\mathrm{R}_{SSI} := \{(y, E_Y) | y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle\}$$

Presig :

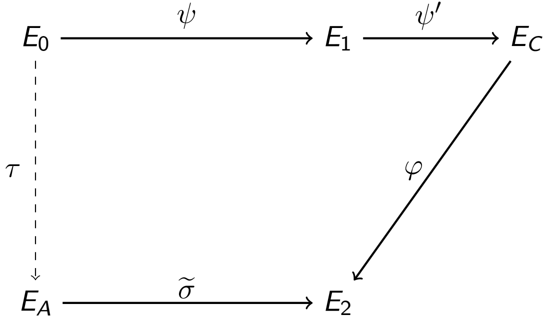


SQISign Adaptor Signature (SAS)

Let (P_0, Q_0) be a basis for $E_0[\ell^e]$, for some small prime ℓ.
We choose our hard relation to be

$$\mathrm{R}_{SSI} := \{(y, E_Y) | y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle\}$$

Presig :

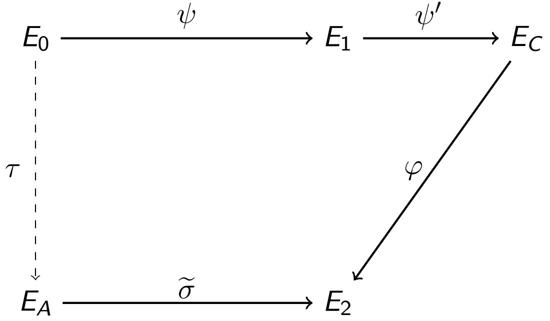


SQISign Adaptor Signature (SAS)

Let (P_0, Q_0) be a basis for $E_0[\ell^e]$, for some small prime ℓ.
We choose our hard relation to be

$$\mathrm{R}_{SSI} := \{(y, E_Y) | y : E_0 \to E_Y \cong E_0 / \langle P_0 + \alpha_y Q_0 \rangle\}$$

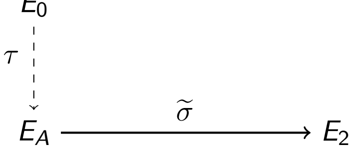
Presig :



Include $\tau(P_0), \tau(Q_0)$ in PreSig

SQISign Adaptor Signature (SAS)

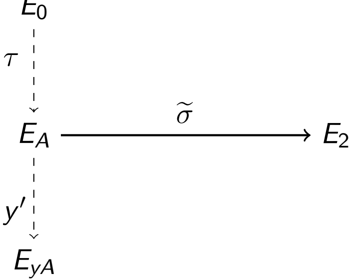
Adapt : (y, E_Y) where $y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle$



SQISign Adaptor Signature (SAS)

Adapt : (y, E_Y) where $y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle$

$$y' : E_A \to E_{yA} = E_A/\langle \tau(P_0) + \alpha_y \tau(Q_0) \rangle$$

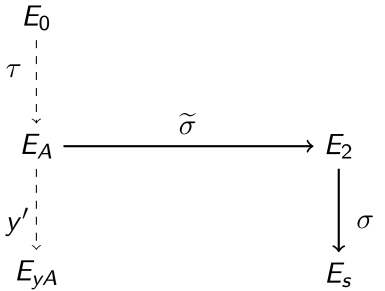


SQISign Adaptor Signature (SAS)

Adapt : (y, E_Y) where $y : E_0 \to E_Y \cong E_0/\langle P_0 + \alpha_y Q_0 \rangle$

$$y' : E_A \to E_{yA} = E_A/\langle \tau(P_0) + \alpha_y \tau(Q_0) \rangle$$

$$\sigma : E_2 \to E_s = E_2/\langle \widetilde{\sigma}(\tau(P_0) + \alpha_y \tau(Q_0)) \rangle$$



SQISign Adaptor Signature (SAS)

Where it went wrong:

- Need a new prime;
- adhoc security assumptions;
- not secure.

Anonymous Multi-Hop Locks (AMHL) via SAS

Setup:

Anonymous Multi-Hop Locks (AMHL) via SAS

Setup:

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathbb{Z}.$$

For each $j \in [1, \cdots k-1]$:

$$\alpha_j = \sum_{i=0}^{j} \ell_i, y_j : E_0 \to E_{Yj} \cong E_0/\langle P_0 + \alpha_j Q_0 \rangle$$

Anonymous Multi-Hop Locks (AMHL) via SAS

Setup:

$$\{\ell_1, \cdots \ell_{k-1}\} \subset \mathbb{Z}.$$

For each $j \in [1, \cdots k-1]$:

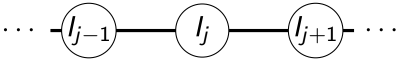
$$\alpha_j = \sum_{i=0}^{j} \ell_i, y_j : E_0 \to E_{Yj} \cong E_0 / \langle P_0 + \alpha_j Q_0 \rangle$$

Commit:

Each I_j will create a pre-signature $\hat{\sigma}_i = \mathsf{PreSig}(\mathsf{sk_i}, \mathsf{tx_i}, \mathsf{E_{Yj}})$ where $\mathsf{tx_i}$ is the conditional contract stating that I_j will release funds to I_{j+1} once I_j is provided their full signature.

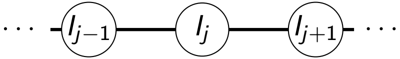
Anonymous Multi-Hop Locks (AMHL) via SAS

Release:



Anonymous Multi-Hop Locks (AMHL) via SAS

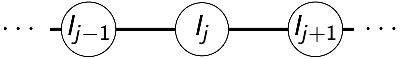
Release:



$$\sigma_j \leftarrow l_{j+1}$$

Anonymous Multi-Hop Locks (AMHL) via SAS

Release:



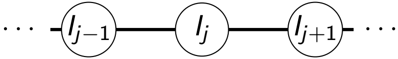
$$\sigma_j \leftarrow l_{j+1}$$

$$y_j \leftarrow \mathsf{Extract}(\sigma_j, \widetilde{\sigma}_j, E_{Yj})$$

$$\alpha_j \leftarrow y_j$$

Anonymous Multi-Hop Locks (AMHL) via SAS

Release:



$$\sigma_j \leftarrow I_{j+1}$$

$$y_j \leftarrow \mathrm{Extract}(\sigma_j, \widetilde{\sigma}_j, E_{Yj})$$

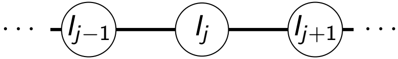
$$\alpha_j \leftarrow y_j$$

$$\alpha_{j-1} = \alpha_j - \ell_j$$

$$\mathrm{y}_{j-1} : E_0 \rightarrow E_{Yj-1} \cong E_0/\langle P_0 + \alpha_{j-1} Q_0 \rangle$$

Anonymous Multi-Hop Locks (AMHL) via SAS

Release:



$$\sigma_j \leftarrow I_{j+1}$$

$$y_j \leftarrow \mathrm{Extract}(\sigma_j, \widetilde{\sigma}_j, E_{Yj})$$

$$\alpha_j \leftarrow y_j$$

$$\alpha_{j-1} = \alpha_j - \ell_j$$

$$\mathsf{y}_{j-1} : E_0 \to E_{Yj-1} \cong E_0 / \langle P_0 + \alpha_{j-1} Q_0 \rangle$$

$$\sigma_{j-1} \leftarrow \mathrm{Adapt}(\widetilde{\sigma}_{j-1}, y_{j-1})$$

Size Comparison in Bytes for 128-bit Security

	LAS	IAS	SAS
public key (bytes)	1472	128 - 2097152	64
presig (bytes)	2701	18327	226
sig (bytes)	3210	263 - 1880	15704

Size Comparison in Bytes for 128-bit Security

	LAS	IAS	SAS
public key (bytes)	1472	128 - 2097152	64
presig (bytes)	2701	18327	226
sig (bytes)	3210	263 - 1880	15704

The smaller presignature sizes in SAS make it better suited for *long* payment channel networks

- longer networks mean a longer set-up phase
- more will need to be transmitted to the participants