Access Recovery and Attestation Using Strong Authenticators

Asynchronous Remote Key Generation & Key Attestation

January 06, 2026

Hugo Nartz
gitlab.com/rv5MDg

Let’s Go Sightseeing

Somewhere in the Bavarian Alps

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation
- CCA1 (Fully) Homomorphic Encryption

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation
- CCA1 (Fully) Homomorphic Encryption
- Isogeny-related ZKPoK

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation
- CCA1 (Fully) Homomorphic Encryption
- Isogeny-related ZKPoK
- Revokable/Tracable Key Attestation

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation
- CCA1 (Fully) Homomorphic Encryption
- Isogeny-related ZKPoK
- Revokable/Tracable Key Attestation
- Updatable KEM

Topics of Interest in UniBW Munich

- Asynchrous Remote Key Generation
- CCA1 (Fully) Homomorphic Encryption
- Isogeny-related ZKPoK
- Revokable/Tracable Key Attestation
- Updatable KEM

How to register and authenticate users securely without relying on passwords?

Web Authentication (WebAuthn)

Web Authentication (WebAuthn)

As a W3C specification

An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Web Authentication (WebAuthn)

As a W3C specification

An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Example of strong authenticator: Yubikey

Web Authentication (WebAuthn)

As a W3C specification

An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Example of strong authenticator: Yubikey

Physical device & cryptographic keys > password 123C4r4mb42026.

Web Authentication (WebAuthn)

As a W3C specification

An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Example of strong authenticator: Yubikey

Physical device & cryptographic keys > password 123C4r4mb42026.

WebAuthn

Web Authentication (WebAuthn)

As a W3C specification

An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Example of strong authenticator: Yubikey

Physical device & cryptographic keys > password 123C4r4mb42026.

WebAuthn + Client-To-Authenticator Protocol

Web Authentication (WebAuthn)

As a W3C specification

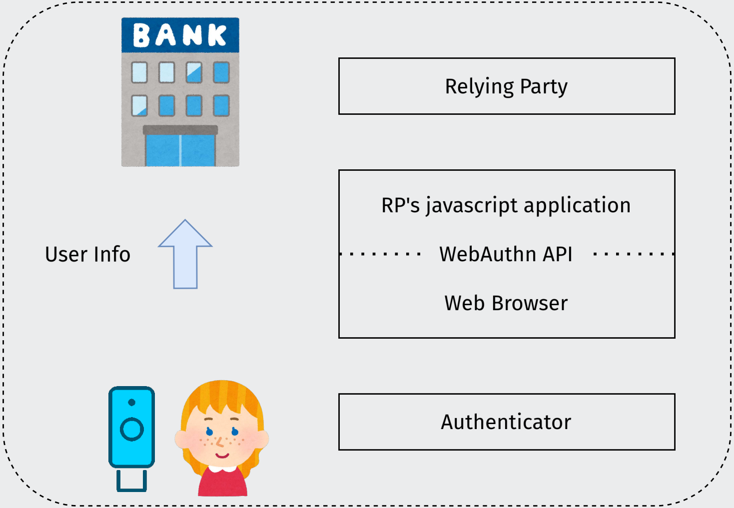
An API allowing servers to register and authenticate users using public key cryptography instead of passwords.

Example of strong authenticator: Yubikey

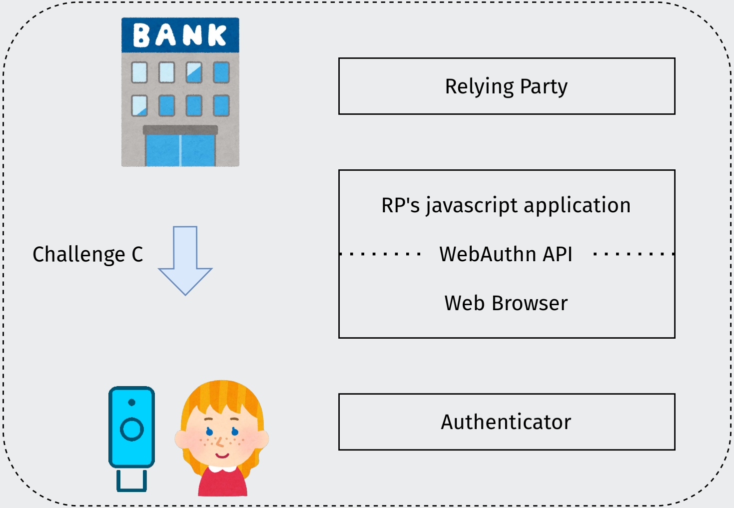
Physical device & cryptographic keys > password 123C4r4mb42026.

WebAuthn + Client-To-Authenticator Protocol = FIDO2.

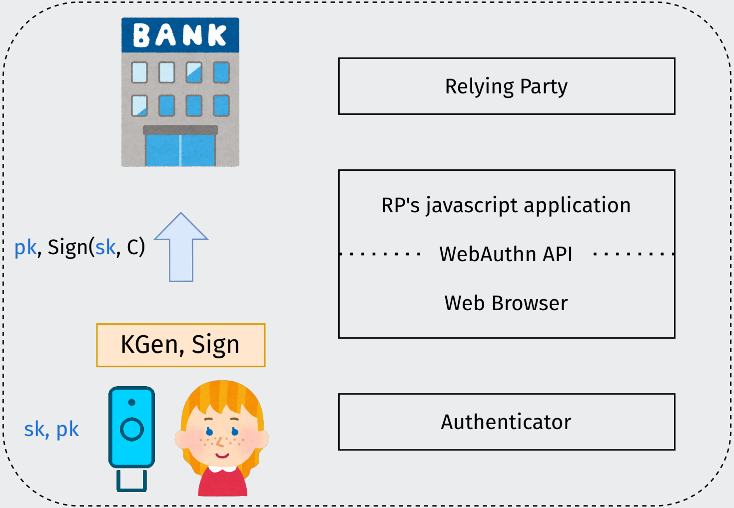
WebAuthn: registration



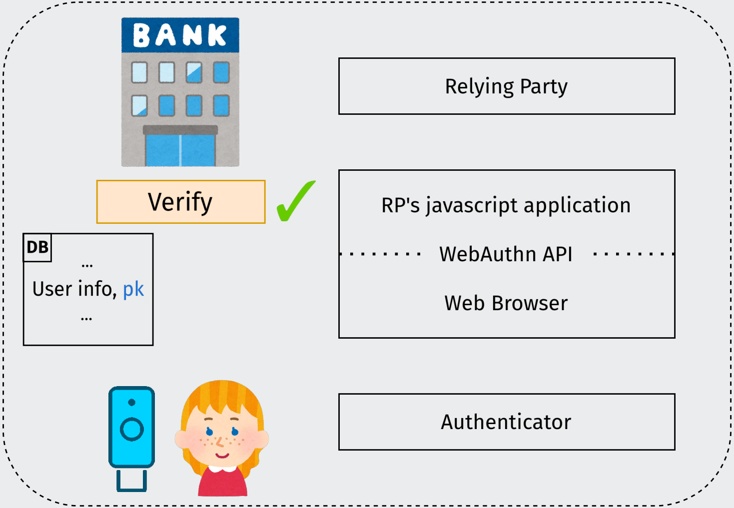
WebAuthn: registration



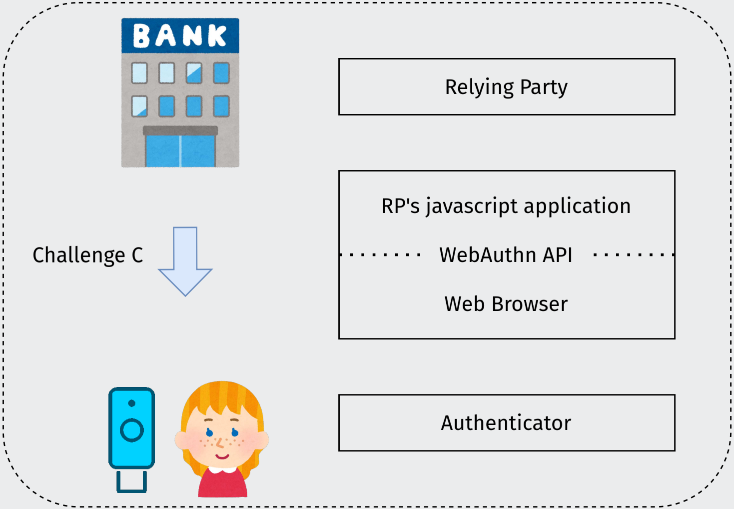
WebAuthn: registration



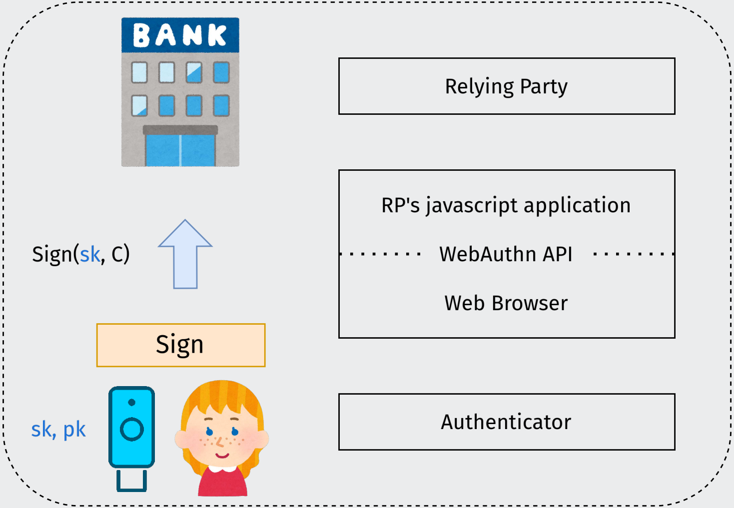
WebAuthn: registration



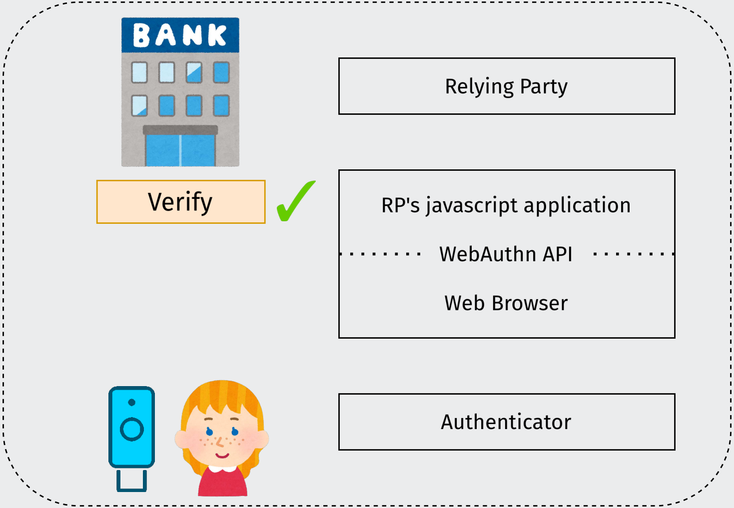
WebAuthn: authentication



WebAuthn: authentication



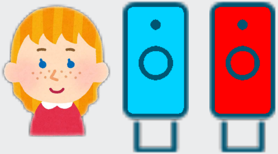
WebAuthn: authentication



Dealing with authenticator loss

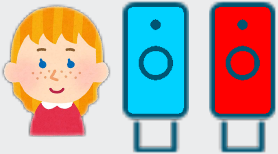
Dealing with authenticator loss

Solution: always carry a backup authenticator and register both

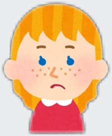


Dealing with authenticator loss

Solution: always carry a backup authenticator and register both



They may be lost simultaneously.



Dealing with authenticator loss

Better solution: Asynchronous Remote Key Generation (ARKG).

Introduced at ACM CCS 2020 by Frymann et al.

Multiple authenticators, only one is used during registration.

Applications

Where unlinkability and asynchrony are required.

Examples

- WebAuthn/FIDO2
- Unlinkable delegation of accounts
- Stealth addresses and signatures
- Anonymous encryption/KEM

Standardization: IETF draft currently being written.

Asynchronous Remote Key Generation (ARKG)

Setting: main and the backup or proxy.

Asynchronous Remote Key Generation (ARKG)

Setting: main and the backup or proxy.

Syntax

1. $\mathsf{KGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$

Asynchronous Remote Key Generation (ARKG)

Setting: main and the backup or proxy.

Syntax

1. $\mathtt{KGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$
2. $\mathtt{DerivePK}(\mathsf{pk}) \rightarrow (\mathsf{pk}', \mathsf{cred}) \rightsquigarrow$ stored on server under $(\text{User Info}, \mathsf{pk}, (\mathsf{pk}', \mathsf{cred}))$

Asynchronous Remote Key Generation (ARKG)

Setting: main and the backup or proxy.

Syntax

1. $\mathtt{KGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$
2. $\mathtt{DerivePK}(\mathsf{pk}) \rightarrow (\mathsf{pk}', \mathsf{cred}) \rightsquigarrow$ stored on server under $\big(\text{User Info}, \mathsf{pk}, (\mathsf{pk}', \mathsf{cred})\big)$
3. $\mathtt{DeriveSK}(\mathsf{sk}, \mathsf{cred}) \rightarrow \mathsf{sk}'$

Asynchronous Remote Key Generation (ARKG)

Setting: main and the backup or proxy.

Syntax

1. $\mathsf{KGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$
2. $\mathsf{DerivePK}(\mathsf{pk}) \rightarrow (\mathsf{pk}', \mathsf{cred}) \rightsquigarrow$ stored on server under $\left(\text{User Info}, \mathsf{pk}, (\mathsf{pk}', \mathsf{cred})\right)$
3. $\mathsf{DeriveSK}(\mathsf{sk}, \mathsf{cred}) \rightarrow \mathsf{sk}'$
4. $\mathsf{Check}(\mathsf{sk}, \mathsf{pk}) \rightarrow \top / \bot$

DLog-based ARKG instantiation

Setting: key pairs of the form $(\mathsf{sk}, \mathsf{pk}) = (s, g^s)$, examples: Schnorr, ECDSA, ElGamal.

DLog-based ARKG instantiation

Setting: key pairs of the form $(\mathsf{sk}, \mathsf{pk}) = (s, g^s)$, examples: Schnorr, ECDSA, ElGamal.

$\texttt{DerivePK}(\mathsf{pk})$

1 : $(e, E) \leftarrow \mathsf{KGen}(1^\lambda)$

2 : $k \leftarrow \mathsf{KDF}_1(\mathsf{pk}^e)$

3 : $P \leftarrow g^k \cdot \mathsf{pk}$

4 : **return** $\mathsf{pk}' = P, \mathsf{cred} = E$

$\texttt{DeriveSK}(\mathsf{sk}, \mathsf{cred} = E)$

1 : $k \leftarrow \mathsf{KDF}_1(E^{\mathsf{sk}})$

2 : **return** $\mathsf{sk}' = k + \mathsf{sk}$

General(?) ARKG instantiation (using KEM)

Setting: key pairs $(\mathsf{sk}_\Delta, \mathsf{pk}_\Delta)$ for a (signature) scheme Δ and $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi)$ for a KEM Π.

General(?) ARKG instantiation (using KEM)

Setting: key pairs $(\mathsf{sk}_\Delta, \mathsf{pk}_\Delta)$ for a (signature) scheme Δ and $(\mathsf{sk}_\Pi, \mathsf{pk}_\Pi)$ for a KEM Π.

$\mathtt{DerivePK}(\mathsf{pk} = (\mathsf{pk}_\Delta, \mathsf{pk}_\Pi))$

1 : $(K, \mathsf{ct}) \leftarrow \mathtt{KEM.Encaps}(\mathsf{pk}_\Pi)$

2 : $k \leftarrow \mathsf{KDF}_1(K)$

3 : $P \leftarrow \mathtt{BlindPK}(\mathsf{pk}_\Delta, k)$

4 : **return** $\mathsf{pk}'_\Delta = P, \mathsf{cred} = \mathsf{ct}$

$\mathtt{DeriveSK}(\mathsf{sk} = (\mathsf{sk}_\Delta, \mathsf{sk}_\Pi), \mathsf{cred} = \mathsf{ct})$

1 : $K \leftarrow \mathtt{KEM.Decaps}(\mathsf{sk}_\Pi, \mathsf{ct})$

2 : $k \leftarrow \mathsf{KDF}_1(K)$

3 : **return** $\mathsf{sk}'_\Delta = \mathtt{BlindSK}(\mathsf{sk}_\Delta, k)$

Security of ARKG

Security of ARKG

Must follow the WebAuthn requirements.

Security of ARKG

Must follow the WebAuthn requirements.

Secret-Key Secrecy

An adversary cannot generate valid $(\mathsf{sk}', \mathsf{pk}', \mathsf{cred})$.

Multiple variants: honest/malicious and weak/strong.

$$ms \Rightarrow mw \Rightarrow hw \text{ and } ms \Rightarrow hs \Rightarrow hw.$$

Security of ARKG

Must follow the WebAuthn requirements.

Secret-Key Secrecy

An adversary cannot generate valid $(\mathsf{sk}', \mathsf{pk}', \mathsf{cred})$.

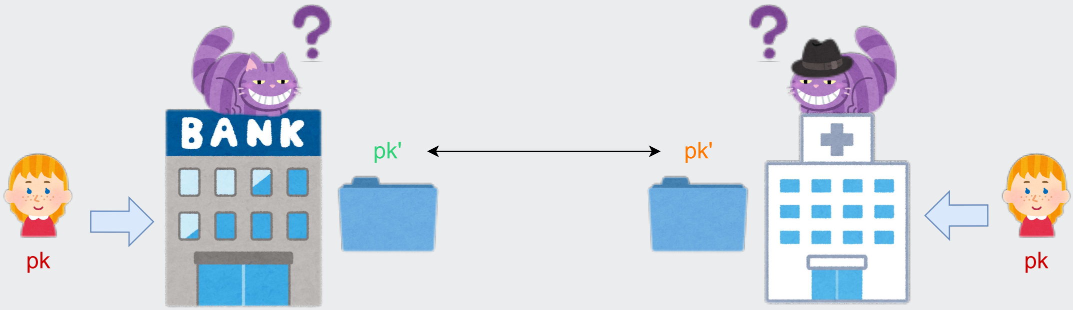
Multiple variants: honest/malicious and weak/strong.

$$ms \Rightarrow mw \Rightarrow hw \text{ and } ms \Rightarrow hs \Rightarrow hw.$$

Public-Key Unlinkability

An adversary with access to pk cannot tell derived keys pk' from freshly generated ones.

PK-Unlinkability



Current State of ARKG: Existing Schemes

Discrete Logarithm and Bilinear Keys

- [FGKLMN20] original Dlog-based scheme.
- [FGMN23] general framework for pairings.
- [MN25] distributed ARKG.

Targeting Dilithium signature scheme (lattice-based)

- [FGM23] based on split-KEMs and rejection sampling (Kyber).
- [BCF23] using only KEM to share randomness (Kyber, fully trusted delegator)

Targeting variants of CSI-FiSh, Dilithium and LegRoast

- [W23] also uses Kyber, focuses on blinding schemes.

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \mathtt{BlindPK}(\mathsf{pk}, k)$.

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \texttt{BlindPK}(\mathsf{pk}, k)$.
For DLog: $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot g^k$ or pk^k.

Digression: An Annoying Obstruction

Blinding keys: $\text{pk}' \leftarrow \texttt{BlindPK}(\text{pk}, k)$.
For DLog: $\text{pk}' \leftarrow \text{pk} \cdot g^k$ or pk^k.
For isogenies: $\text{pk}' \leftarrow \phi_k \cdot \text{pk}$.

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \texttt{BlindPK}(\mathsf{pk}, k)$.
For DLog: $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot g^k$ or pk^k.
For isogenies: $\mathsf{pk}' \leftarrow \phi_k \cdot \mathsf{pk}$.
For lattices:

$$\mathsf{pk}' \leftarrow (A, (A \cdot \mathbf{sk} + \mathbf{e}) + (A \cdot \mathbf{k})) = (A, A \cdot (\mathbf{sk} + \mathbf{k}) + \mathbf{e})$$

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \mathtt{BlindPK}(\mathsf{pk}, k)$.
For DLog: $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot g^k$ or pk^k.
For isogenies: $\mathsf{pk}' \leftarrow \phi_k \cdot \mathsf{pk}$.
For lattices:

$$\mathsf{pk}' \leftarrow (A, (A \cdot \mathbf{sk} + \mathbf{e}) + (A \cdot \mathbf{k})) = (A, A \cdot (\mathbf{sk} + \mathbf{k}) + \mathbf{e})$$

with $\mathbf{e}$ and $\mathbf{sk}$ vectors sampled from Gaussian distributions.

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \texttt{BlindPK}(\mathsf{pk}, k)$.

For DLog: $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot g^k$ or pk^k.

For isogenies: $\mathsf{pk}' \leftarrow \phi_k \cdot \mathsf{pk}$.

For lattices:

$$\mathsf{pk}' \leftarrow (A, (A \cdot \mathbf{sk} + \mathbf{e}) + (A \cdot \mathbf{k})) = (A, A \cdot (\mathbf{sk} + \mathbf{k}) + \mathbf{e})$$

with $\mathbf{e}$ and $\mathbf{sk}$ vectors sampled from Gaussian distributions.

What about the distribution of sk'?

Digression: An Annoying Obstruction

Blinding keys: $\mathsf{pk}' \leftarrow \texttt{BlindPK}(\mathsf{pk}, k)$.
For DLog: $\mathsf{pk}' \leftarrow \mathsf{pk} \cdot g^k$ or pk^k.
For isogenies: $\mathsf{pk}' \leftarrow \phi_k \cdot \mathsf{pk}$.
For lattices:

$$\mathsf{pk}' \leftarrow (A, (A \cdot \mathbf{sk} + \mathbf{e}) + (A \cdot \mathbf{k})) = (A, A \cdot (\mathbf{sk} + \mathbf{k}) + \mathbf{e})$$

with $\mathbf{e}$ and $\mathbf{sk}$ vectors sampled from Gaussian distributions.
What about the distribution of sk'?

$$\mathcal{G}_\sigma + \mathcal{G}_\sigma \sim \mathcal{G}_{\sqrt{2}\sigma}$$

So sk' is not distributed in the same way as a fresh key.

Solution: Rejection Sampling

Security of the Dlog-based instantiation

Secret-Key Secrecy

Honest-strong $\Leftarrow$ Dlog assumption in standard model.

Malicious-strong $\Leftarrow$ snPRF-ODH assumption in the ROM.

Public-Key Unlinkability

Follows from the nnPRF-ODH assumption in the ROM.

Security of the Dlog-based instantiation

Secret-Key Secrecy

Honest-strong $\Leftarrow$ Dlog assumption in standard model.

Malicious-strong $\Leftarrow$ snPRF-ODH assumption in the ROM.

Public-Key Unlinkability

Follows from the nnPRF-ODH assumption in the ROM.

snPRF-ODH: introduced to study TLS1.3.

Extension to Pairings: Generic Security Properties

Extension to Pairings: Generic Security Properties

ϕ-AKG: An asymmetric scheme together with a map $\phi : \mathsf{G}_{\mathsf{sk}} \times \mathsf{G}_{\mathsf{pk}} \to \mathsf{G}_{\mathsf{pk}}$.

Extension to Pairings: Generic Security Properties

ϕ-AKG: An asymmetric scheme together with a map $\phi : \mathsf{G}_{\mathsf{sk}} \times \mathsf{G}_{\mathsf{pk}} \to \mathsf{G}_{\mathsf{pk}}$.
Let KDF_1 (and KDF_2, MAC) be a secure function.

Theorem (msKS/mwKS-Secret-Key Secrecy)

If (ϕ-AKG, KDF_1) is secure under the snPRF-O_ϕ assumption, the compiled ARKG scheme is msKS-*secure (and therefore* mwKS-*secure).*

Theorem (Public-Key Unlinkability)

If (ϕ-AKG, KDF_1) is secure under the nnPRF-O_ϕ *assumption, the compiled ARKG scheme satisfies PK-unlinkability.*

Instantiation: Bilinear Groups

Instantiation: Bilinear Groups

A description of a bilinear group $\mathcal{G}$ is a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \gamma, p)$ such that

- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p,
- $\mathbb{G}_1$ (*resp.* $\mathbb{G}_2$) is generated by element g_1 (*resp.* g_2),
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear pairing,
- $\gamma : \mathbb{G}_2 \to \mathbb{G}_1$ is an isomorphism.

$$e(g_1^a, g_2^b) = e(g_1, g_2^b)^a = e(g_1^a, g_2)^b = e(g_1, g_2)^{ab}.$$

Instantiation: Bilinear Groups

A description of a bilinear group $\mathcal{G}$ is a tuple $(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, e, \gamma, p)$ such that

- $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_T$ are cyclic groups of prime order p,
- $\mathbf{G}_1$ (*resp.* $\mathbf{G}_2$) is generated by element g_1 (*resp.* g_2),
- $e : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$ is a non-degenerate bilinear pairing,
- $\gamma : \mathbf{G}_2 \to \mathbf{G}_1$ is an isomorphism.

$$e(g_1^a, g_2^b) = e(g_1, g_2^b)^a = e(g_1^a, g_2)^b = e(g_1, g_2)^{ab}.$$

Assumptions on $\mathbf{G}_1$ and $\mathbf{G}_2$ (CDH, DDH, ...) and on the efficient computability of γ/γ^{-1} (the type of $\mathcal{G}$): XDH, SXDH, DBDH, ...

Building ϕ and Unlinkability with a Pairing

Asymmetric keys parametrized by exponent vectors: $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x}))$ with $\vec{x} \in \mathbb{Z}_p^{n_1+n_2+n_T}$.

Building ϕ and Unlinkability with a Pairing

Asymmetric keys parametrized by exponent vectors: $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x}))$ with $\vec{x} \in \mathbb{Z}_p^{n_1+n_2+n_T}$.

Example: Type-1 group with $\vec{x} = (x_1, x_2)$ and $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x})) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$

Building ϕ and Unlinkability with a Pairing

Asymmetric keys parametrized by exponent vectors: $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x}))$ with $\vec{x} \in \mathbb{Z}_p^{n_1+n_2+n_T}$.

Example: Type-1 group with $\vec{x} = (x_1, x_2)$ and $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x})) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$

Under nnPRF-$\mathcal{O}_\phi$:

$$\begin{aligned} \text{PK-Unlinkability} &\Leftarrow \mathsf{PRF}(\phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y}))) \sim z \leftarrow\$ \mathbf{G}_{\mathsf{sk}} \\ &\Leftrightarrow \phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) \sim Z' \leftarrow\$ \mathbf{G}_T. \end{aligned}$$

Building ϕ and Unlinkability with a Pairing

Asymmetric keys parametrized by exponent vectors: $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x}))$ with $\vec{x} \in \mathbb{Z}_p^{n_1+n_2+n_T}$.
Example: Type-1 group with $\vec{x} = (x_1, x_2)$ and $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x})) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$
Under nnPRF-$\mathcal{O}_\phi$:

$$\text{PK-Unlinkability} \Leftarrow \text{PRF}(\phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y}))) \sim z \leftarrow\$ \mathbf{G}_{\mathsf{sk}}$$
$$\Leftrightarrow \phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) \sim Z' \leftarrow\$ \mathbf{G}_T.$$

Mapping ϕ for Camenisch-Lysyanskaya signatures

Bilinear group $\mathcal{G}$ of type 1: $e : \mathbb{G} \times \mathbb{G} \to \mathbf{G}_T$

$$\phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) = \phi((x_1, x_2), (g^{y_1}, g^{y_2})) := e(g^{y_1}, g^{y_2})^{x_1 x_2} = g_T^{x_1 x_2 y_1 y_2}.$$

Building ϕ and Unlinkability with a Pairing

Asymmetric keys parametrized by exponent vectors: $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x}))$ with $\vec{x} \in \mathbb{Z}_p^{n_1+n_2+n_T}$.

Example: Type-1 group with $\vec{x} = (x_1, x_2)$ and $(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{x})) = ((x_1, x_2), (g^{x_1}, g^{x_2}))$

Under nnPRF-$\mathcal{O}_\phi$:

$$\text{PK-Unlinkability} \Leftarrow \mathsf{PRF}(\phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) \sim z \leftarrow\$ \mathbf{G}_{\mathsf{sk}}$$
$$\Leftrightarrow \phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) \sim Z' \leftarrow\$ \mathbf{G}_T.$$

Mapping ϕ for Camenisch-Lysyanskaya signatures

Bilinear group $\mathcal{G}$ of type 1: $e : \mathbb{G} \times \mathbb{G} \to \mathbf{G}_T$

$$\phi(\mathsf{sk}(\vec{x}), \mathsf{pk}(\vec{y})) = \phi((x_1, x_2), (g^{y_1}, g^{y_2})) := e(g^{y_1}, g^{y_2})^{x_1 x_2} = g_T^{x_1 x_2 y_1 y_2}.$$

$$\text{PK-Unlinkability} \Leftarrow g_T^{Q(\vec{x}, \vec{y})} \sim Z' \leftarrow\$ \mathbf{G}_T.$$

Reduction to decisional UBER-assumption familly

Parametrization for Camenisch-Lysyanskaya signatures

4-multivariate polynomial vectors: $\vec{F}, \vec{H}, \vec{K}$ in X_1, X_2, Y_1, Y_2:

$$\vec{F} = (X_1, X_2), \vec{H} = (Y_1, Y_2), \vec{K} = \varnothing$$
$$Q(X_1, X_2, Y_1, Y_2) = X_1 X_2 Y_1 Y_2.$$

Reduction to decisional UBER-assumption familly

Parametrization for Camenisch-Lysyanskaya signatures

4-multivariate polynomial vectors: $\vec{F}, \vec{H}, \vec{K}$ in X_1, X_2, Y_1, Y_2:

$$\vec{F} = (X_1, X_2), \vec{H} = (Y_1, Y_2), \vec{K} = \varnothing$$
$$Q(X_1, X_2, Y_1, Y_2) = X_1 X_2 Y_1 Y_2.$$

$(\vec{F}, \vec{H}, \vec{K}, Q)$-Decisional UBER experiment

Given $g_1^{\vec{F}(\vec{x},\vec{y})}$, $g_2^{\vec{H}(\vec{x},\vec{y})}$ and $g_T^{\vec{K}(\vec{x},\vec{y})}$, distinguish $g_T^{Q(\vec{x},\vec{y})}$ from random sampling on $\mathbf{G}_T$.

Reduction to decisional UBER-assumption familly

Parametrization for Camenisch-Lysyanskaya signatures

4-multivariate polynomial vectors: $\vec{F}, \vec{H}, \vec{K}$ in X_1, X_2, Y_1, Y_2:

$$\vec{F} = (X_1, X_2), \vec{H} = (Y_1, Y_2), \vec{K} = \varnothing$$
$$Q(X_1, X_2, Y_1, Y_2) = X_1 X_2 Y_1 Y_2.$$

$(\vec{F}, \vec{H}, \vec{K}, Q)$-Decisional UBER experiment

Given $g_1^{\vec{F}(\vec{x},\vec{y})}$, $g_2^{\vec{H}(\vec{x},\vec{y})}$ and $g_T^{\vec{K}(\vec{x},\vec{y})}$, distinguish $g_T^{Q(\vec{x},\vec{y})}$ from random sampling on $\mathbf{G}_T$.

Read: given $\mathrm{pk}(\vec{x})$ and $E = \mathrm{pk}(\vec{y})$, distinguish $\phi(e = \mathrm{sk}(\vec{y}), \mathrm{pk}(\vec{x}))$ from random sampling on $\mathbf{G}_T$ ($\Rightarrow$ PK-Unlinkability by a reduction result).

Concrete assumption for CL: DBDH

DBDH experiment

Given (g^x, g^y, g^z) with $(x, y, z) \leftarrow\!\!\$\ \mathbb{Z}_p^3$ distinguish g_T^{xyz} from random sampling on G_T.

Concrete assumption for CL: DBDH

DBDH experiment

Given (g^x, g^y, g^z) with $(x, y, z) \leftarrow\$ \mathbb{Z}_p^3$ distinguish g_T^{xyz} from random sampling on $\mathbb{G}_T$.

DBDH $\Rightarrow$ Decisional $(\vec{F}, \vec{H}, \vec{K}, Q)$-Decisional UBER assumption $\Rightarrow$ PK-Unlinkability.

More Complicated Keys

Waters signature scheme (type 1):

$$\mathsf{sk} = (g_1^{x_1}, x_2, \ldots, x_l),\ \mathsf{pk} = (g_T^{x_1}, g_2^{x_2}, \ldots, g_2^{x_l})$$

The mapping ϕ used:

$$(g_1^{x_1}, x_2, \ldots, x_l), (g_T^{y_1}, g_2^{y_2}, \ldots, g_2^{y_l}) \mapsto (g_T^{y_1})^{x_1} e(g_1^{x_1}, g_2^{y_2}) e(g_1^{x_2}, g_2^{y_2}) \cdots e(g_1^{x_l}, g_2^{y_l}).$$

Results: Instantiations of ARKG for Pairing-Based Signature Schemes

Type-1 (DBDH assumption)

- BLS-1 (trusted CRS)
- Camenisch-Lysyanskaya

Type-3 (SXDH assumption)

- BLS-3
- Pointcheval-Sanders
- SPS-EQ
- Waters

Type-1 ($((X_1, Y_1), \varnothing, \varnothing, X_1Y_1(X_1 + Y_1))$-UBER)

- BLS-1

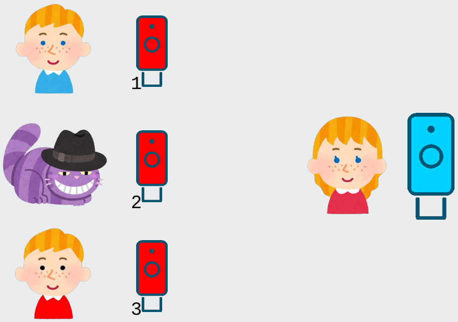
Performances

Table 1: Mean time in milliseconds for each ARKG algorithm. BLS-1/3 and CL are written in C while PS, SPS-EQ and Waters are implemented in python.

	DerivePK	DeriveSK	Check	ARKG total	AKG.KGen
BLS-1	3.56	1.07	0.63	5.26	0.63
BLS-3	2.92	0.99	0.62	4.53	0.61
CL	5.36	0.89	2.21	6.26	2.24
PS	99.23	8.29	0.89	107.52	0.94
SPS-EQ	123.34	17.13	10.89	140.47	5.62
Waters	127.40	17.12	11.52	144.52	8.96

https://gitlab.surrey.ac.uk/sccs/bp-arkg

How to backup access to more than one proxy in a thresholded manner?



The Trivial Case: N-out-of-N Threshold

$\texttt{DerivePK}((\mathsf{pk}_1 = g^{s_1}, \ldots, \mathsf{pk}_N = g^{s_N}))$

1 : $\mathsf{pk} \leftarrow \mathsf{pk}_1 \cdots \mathsf{pk}_N = g^{\sum_i s_i}$

2 : $(e, E) \leftarrow \texttt{KGen}$

3 : $k \leftarrow \mathsf{KDF}_1(\mathsf{pk}^e)$

4 : $P \leftarrow g^k \cdot \mathsf{pk}$

5 : **return** $\mathsf{pk}' = P, \mathsf{cred} = E$

$\texttt{DeriveSK}((\mathsf{sk}_1 = s_1, \cdots, \mathsf{sk}_N = s_N), \mathsf{cred} = E)$

1 : $\mathsf{sk} \leftarrow \sum_i s_i$

2 : $k \leftarrow \mathsf{KDF}_1(E^{\mathsf{sk}})$

3 : **return** $\mathsf{sk}' = k + \mathsf{sk}$

The Trivial Case: N-out-of-N Threshold

$\texttt{DerivePK}((\mathsf{pk}_1 = g^{s_1}, \ldots, \mathsf{pk}_N = g^{s_N}))$

$\mathsf{pk} \leftarrow \mathsf{pk}_1 \cdots \mathsf{pk}_N = g^{\sum_i s_i}$

$(e, E) \leftarrow \texttt{KGen}$

$k \leftarrow \mathsf{KDF}_1(\mathsf{pk}^e)$

$P \leftarrow g^k \cdot \mathsf{pk}$

return $\mathsf{pk}' = P, \mathsf{cred} = E$

$\texttt{DeriveSK}((\mathsf{sk}_1 = s_1, \cdots, \mathsf{sk}_N = s_N), \mathsf{cred} = E)$

$\mathsf{sk} \leftarrow \sum_i s_i$

$k \leftarrow \mathsf{KDF}_1(E^{\mathsf{sk}})$

return $\mathsf{sk}' = k + \mathsf{sk}$

Non-interactive 2-out-of-N ARKG: hard but possible with pairings.

The Trivial Case: N-out-of-N Threshold

$$\underline{\texttt{DerivePK}((\mathsf{pk}_1 = g^{s_1}, \ldots, \mathsf{pk}_N = g^{s_N}))}$$

$\mathsf{pk} \leftarrow \mathsf{pk}_1 \cdots \mathsf{pk}_N = g^{\sum_i s_i}$

$(e, E) \leftarrow \texttt{KGen}$

$k \leftarrow \mathsf{KDF}_1(\mathsf{pk}^e)$

$P \leftarrow g^k \cdot \mathsf{pk}$

$\textbf{return } \mathsf{pk}' = P, \mathsf{cred} = E$

$$\underline{\texttt{DeriveSK}((\mathsf{sk}_1 = s_1, \cdots, \mathsf{sk}_N = s_N), \mathsf{cred} = E)}$$

$\mathsf{sk} \leftarrow \sum_i s_i$

$k \leftarrow \mathsf{KDF}_1(E^{\mathsf{sk}})$

$\textbf{return } \mathsf{sk}' = k + \mathsf{sk}$

Non-interactive 2-out-of-N ARKG: hard but possible with pairings.

Non-interactive 1-out-of-N ARKG $\Rightarrow$ MP-NIKE $\Rightarrow$? iO, multilinear maps.

dARKG: General Goal and Context

Multiple Proxies (backups) chosen by one Delegator (main).

dARKG: General Goal and Context

Multiple Proxies (backups) chosen by one Delegator (main).

Observations

- Proxy/Delegator interactivity is not an issue.
- Proxy/Proxy interactivity unwanted.

dARKG: General Goal and Context

Multiple Proxies (backups) chosen by one Delegator (main).

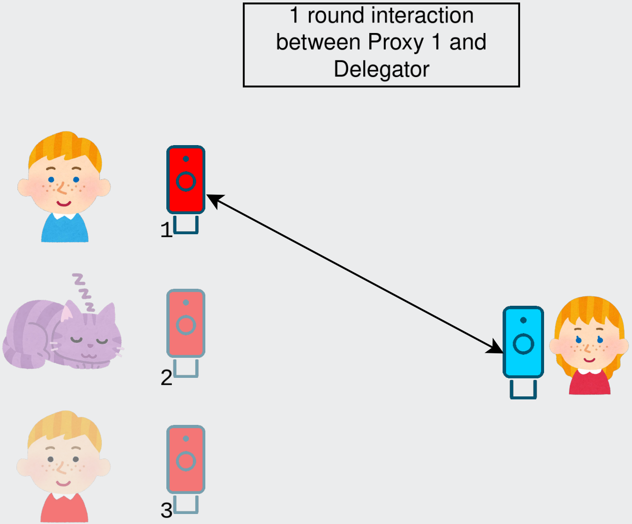
Observations

- Proxy/Delegator interactivity is not an issue.
- Proxy/Proxy interactivity unwanted.

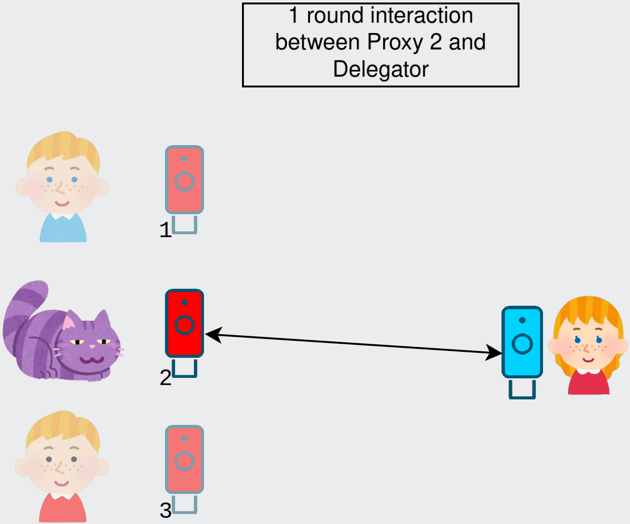
Solution:

- 1-Round Publicly Verifiable Asymmetric Key Agreement (1PVAKA).
- Blinding scheme.
- Threshold secret sharing.

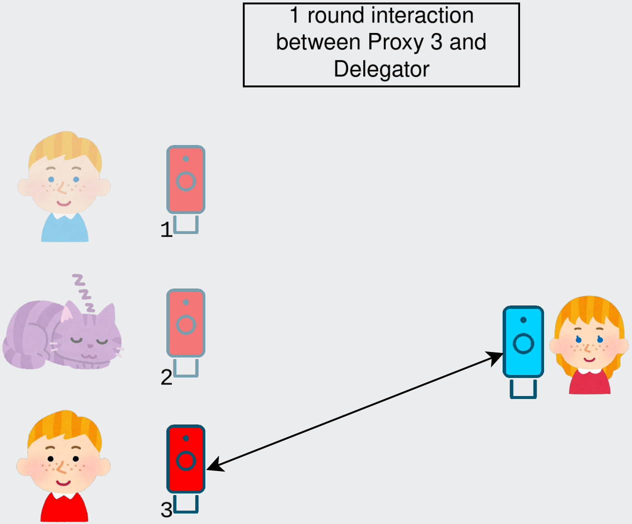
Asymmetric Key Agreement: Generation



Asymmetric Key Agreement: Generation

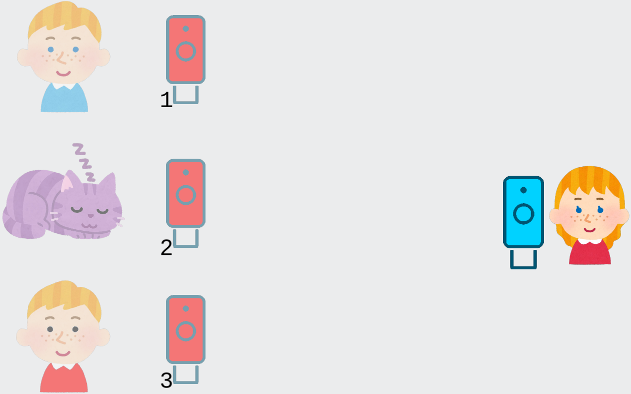


Asymmetric Key Agreement: Generation



Asymmetric Key Agreement: Aggregation

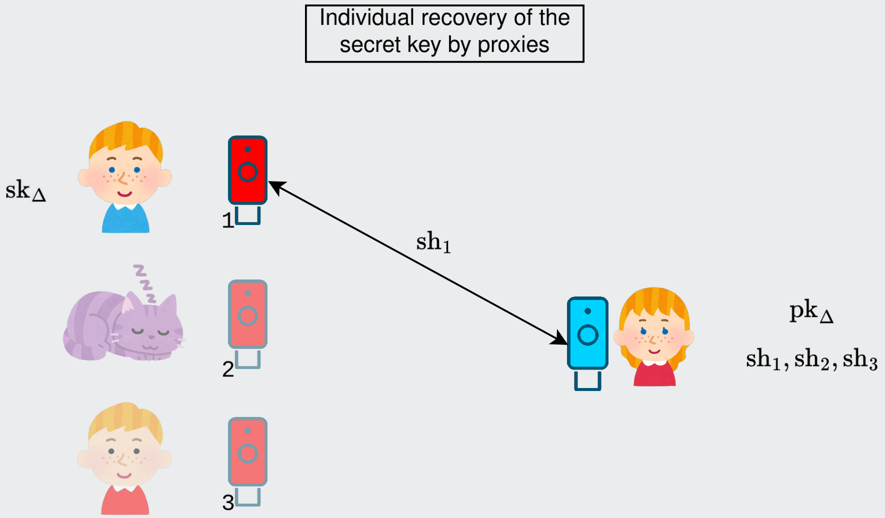
Construction of a shared public key and recovery shares by Delegator



pk_Δ

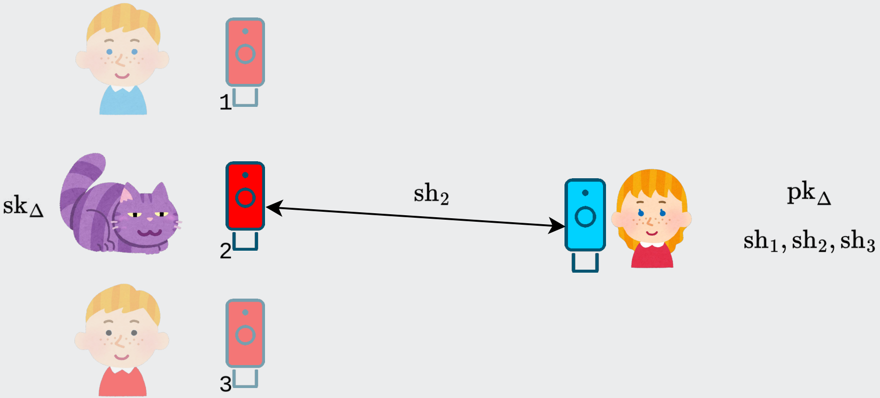
$\mathrm{sh}_1, \mathrm{sh}_2, \mathrm{sh}_3$

Asymmetric Key Agreement: Recovery



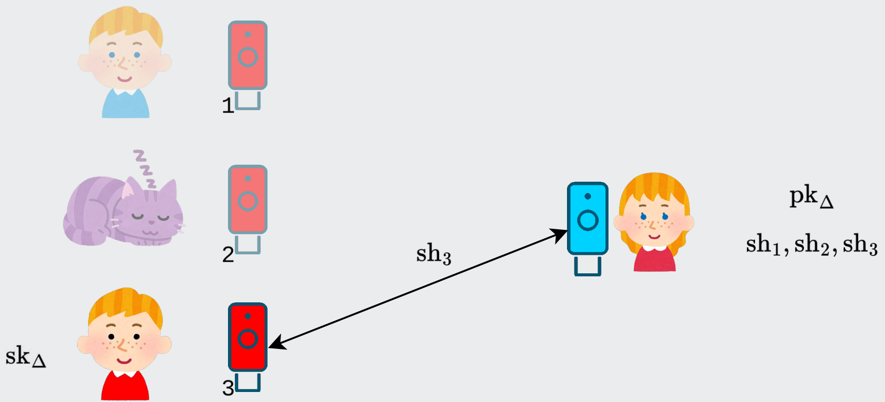
Asymmetric Key Agreement: Recovery

Individual recovery of the secret key by proxies

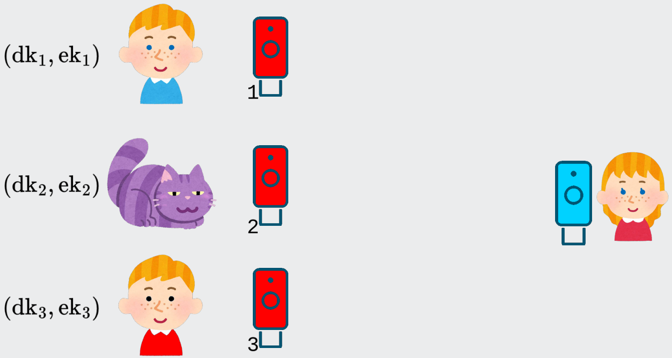


Asymmetric Key Agreement: Recovery

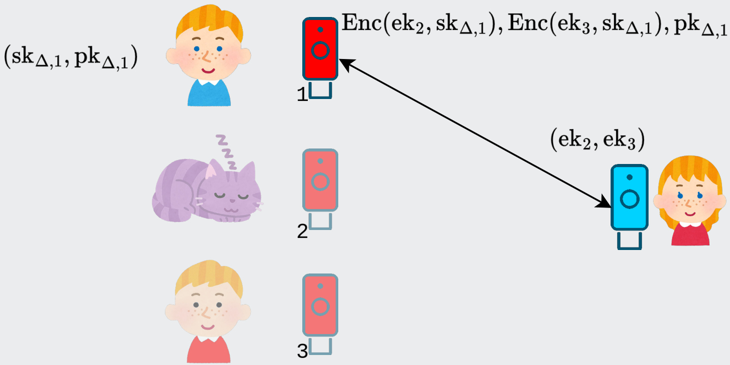
Individual recovery of the secret key by proxies



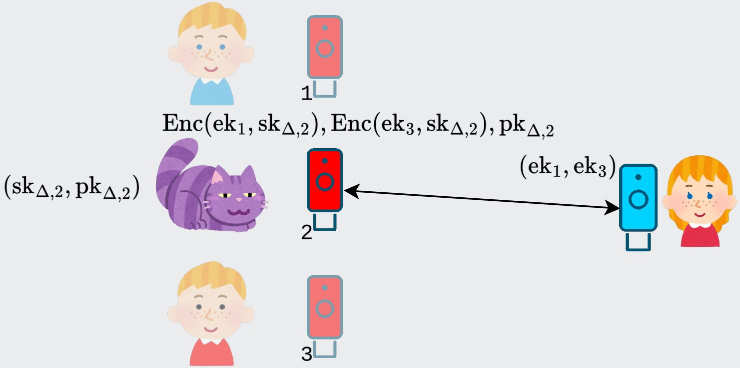
In Practice: Long-Term Encryption Keys



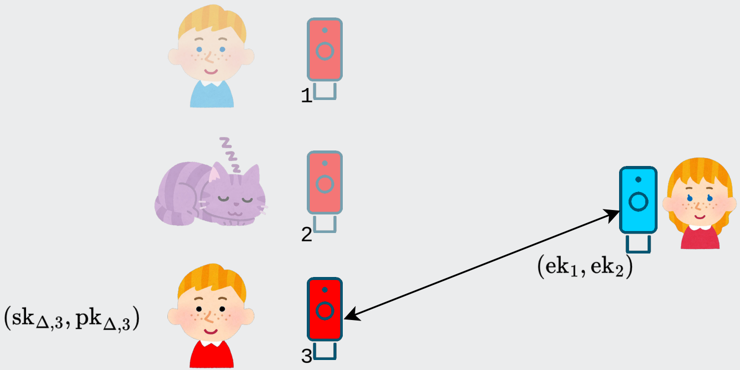
In Practice: Generation



In Practice: Generation



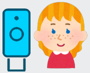
In Practice: Generation



$$\mathrm{Enc}(ek_1, sk_{\Delta,3}), \mathrm{Enc}(ek_2, sk_{\Delta,3}), pk_{\Delta,3}$$

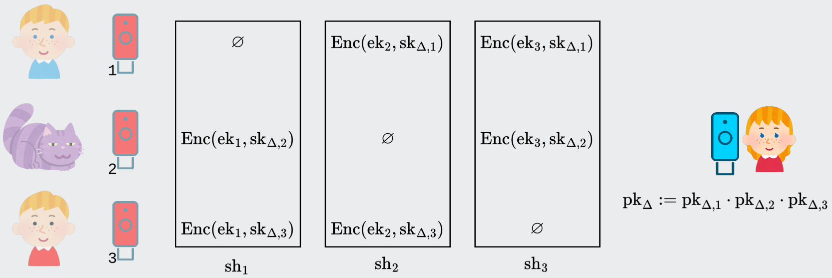
Information sent from each proxy:

$\varnothing$	$\mathrm{Enc}(\mathrm{ek}_2, \mathrm{sk}_{\Delta,1})$	$\mathrm{Enc}(\mathrm{ek}_3, \mathrm{sk}_{\Delta,1})$	$\mathrm{pk}_{\Delta,1}$
$\mathrm{Enc}(\mathrm{ek}_1, \mathrm{sk}_{\Delta,2})$	$\varnothing$	$\mathrm{Enc}(\mathrm{ek}_3, \mathrm{sk}_{\Delta,2})$	$\mathrm{pk}_{\Delta,2}$
$\mathrm{Enc}(\mathrm{ek}_1, \mathrm{sk}_{\Delta,3})$	$\mathrm{Enc}(\mathrm{ek}_2, \mathrm{sk}_{\Delta,3})$	$\varnothing$	$\mathrm{pk}_{\Delta,3}$



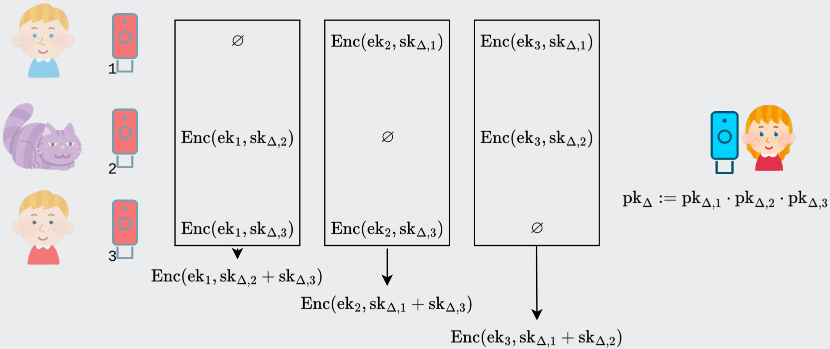
Asymmetric Key Agreement: Aggregation

Aggregation of the public keys and shares by the Delegator:



Asymmetric Key Agreement: Aggregation

Optimization using additively homomorphic encryption:



Recall the General ARKG Algorithms

Setting: key pairs of the form $(\mathsf{sk}_\Delta, \mathsf{pk}_\Delta) = (s, g^s)$.

$$\underline{\texttt{DerivePK}(\mathsf{pk}_\Delta)}$$

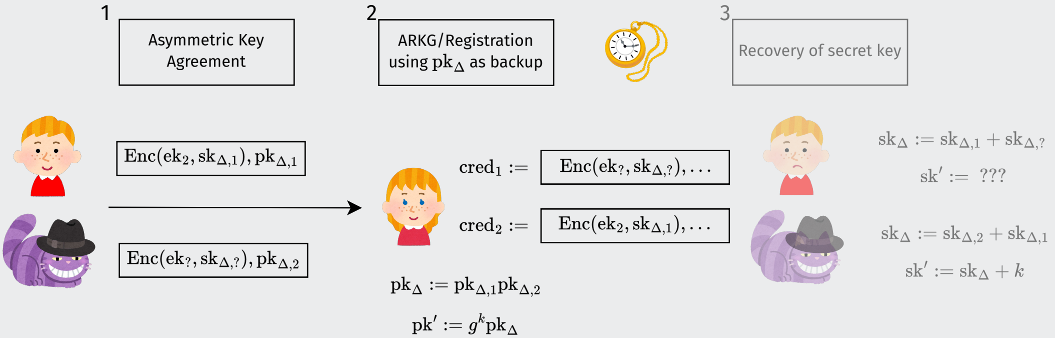
$$\begin{aligned}
&1: \quad (e, E) \leftarrow \texttt{KGen}\\
&2: \quad k \leftarrow \mathsf{KDF}_1(\mathsf{pk}_\Delta^e)\\
&3: \quad P \leftarrow g^k \cdot \mathsf{pk}_\Delta \ \ \textbf{return}\ \mathsf{pk}' = P, \mathsf{cred} = E
\end{aligned}$$

$$\underline{\texttt{DeriveSK}(\mathsf{sk}_\Delta, \mathsf{cred} = E)}$$

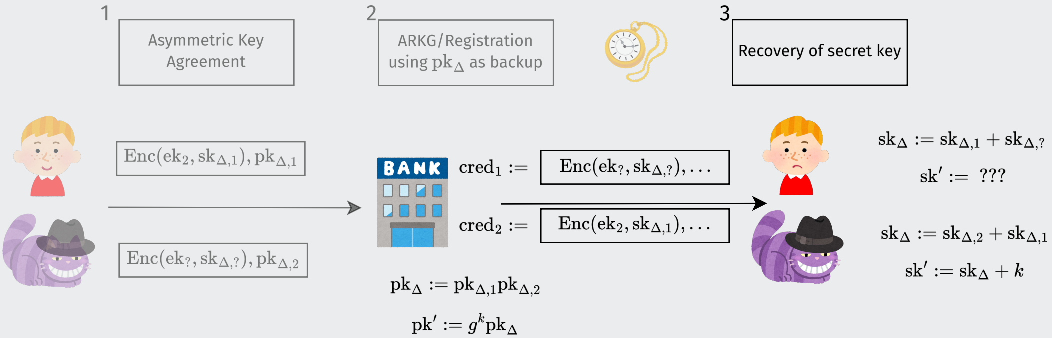
$$\begin{aligned}
&1: \quad k \leftarrow \mathsf{KDF}_1(E^{\mathsf{sk}_\Delta})\\
&2: \quad \textbf{return}\ \mathsf{sk}' = k + \mathsf{sk}_\Delta
\end{aligned}$$

In dARKG, generate pk_Δ using AKA and add the shares to the credentials.
Yields a 1-out-of-N dARKG construction with minimal interactions.

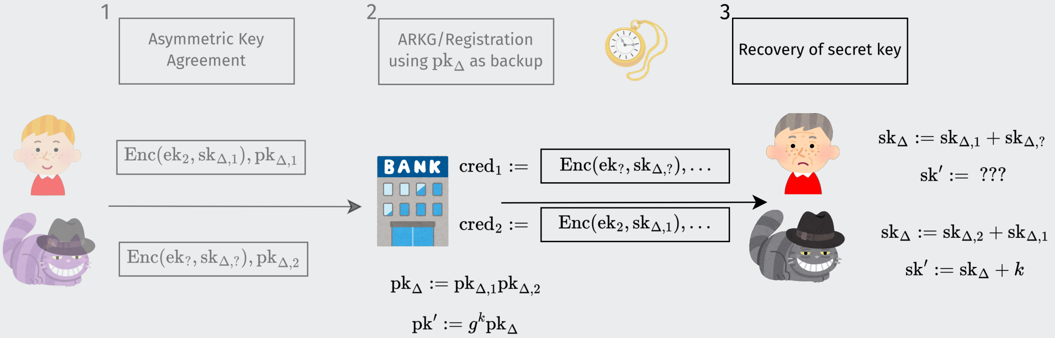
Robustness: Malicious Proxy During AKA Generation



Robustness: Malicious Proxy During AKA Generation



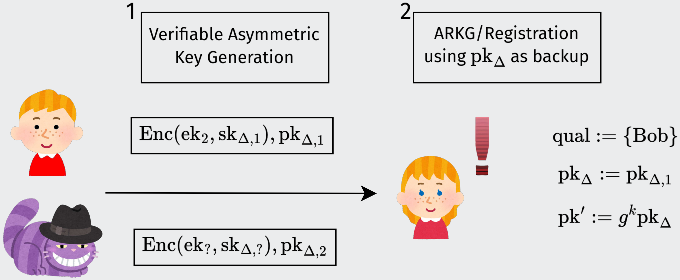
Robustness: Malicious Proxy During AKA Generation



Robustness Using Verifiable Encryption (VE)

Witness/statement relation:

$$(\mathsf{sk}_{\Delta,i})\ \mathcal{R}\ (\mathsf{ek}_i, \mathsf{pk}_{\Delta,i}, \mathsf{ct})$$
$$\Leftrightarrow \mathsf{ct} = \mathsf{Enc}(\mathsf{ek}_i, \mathsf{sk}_{\Delta,i}) \text{ with } g^{\mathsf{sk}_{\Delta i}} = \mathsf{pk}_{\Delta,i}.$$



Decentralized ARKG

Shared key pk_Δ: created via 1-Round Publicly-Verifiable AKA using the delegator as relay.

Decentralized ARKG

Shared key pk_Δ: created via 1-Round Publicly-Verifiable AKA using the delegator as relay.
VE: Multi-recipient encryption + custom NIZK.
Additively homomorphic encryption to compress ciphertexts and thus credentials.

Decentralized ARKG

Shared key pk_Δ**:** created via 1-Round Publicly-Verifiable AKA using the delegator as relay.
VE: Multi-recipient encryption + custom NIZK.
Additively homomorphic encryption to compress ciphertexts and thus credentials.
Blinding k**:** for unlinkability.

Decentralized ARKG

Shared key pk_Δ**:** created via 1-Round Publicly-Verifiable AKA using the delegator as relay.
VE: Multi-recipient encryption + custom NIZK.
Additively homomorphic encryption to compress ciphertexts and thus credentials.
Blinding k**:** for unlinkability.
Threshold k'**:** shared blinding factor created by the delegator.

Decentralized ARKG

Shared key pk_Δ**:** created via 1-Round Publicly-Verifiable AKA using the delegator as relay.
VE: Multi-recipient encryption + custom NIZK.
Additively homomorphic encryption to compress ciphertexts and thus credentials.
Blinding k**:** for unlinkability.
Threshold k'**:** shared blinding factor created by the delegator.
Encrypted for each proxy using standard PKE.

$$\mathsf{pk}_\Delta = \prod\nolimits_{i \in \mathsf{qual}} \mathsf{pk}_{\Delta,i} \qquad \mathsf{pk}' = g^k \cdot g^{k'} \cdot \mathsf{pk}_\Delta$$

$$\mathsf{cred}_i = \{\mathsf{VE.Enc}(\mathsf{ek}_i, \mathsf{sk}_{\Delta,i}), \mathsf{Enc}(\mathsf{ek}_i, \mathsf{sh}_i), \mathsf{MAC}(\ldots), \ldots\}$$

$$\mathsf{sk}' = \sum_{i \in \mathsf{qual}} \mathsf{sk}_{\Delta,i} + k' + k.$$

1PVAKA, Threshold, Blinding.

dARKG: Results and Performances

New syntax and security models for 1PVAKA and dARKG along with generic constructions.
Instantiation based on additive ElGamal and pairing-friendly curve BLS12-381.
`https://gitlab.com/rv5MDg/jupyter-notebook-darkg`

<table>
<tr><td>N, t</td><td>2, 1</td><td>4, 1</td><td>4, 3</td><td>8, 1</td><td>8, 7</td><td>16, 1</td><td>16, 15</td></tr>
<tr><td>KGenProxy</td><td>0.5</td><td colspan="2">1.0</td><td colspan="2">2.1</td><td colspan="2">4.0</td></tr>
<tr><td>DeriveSK</td><td>0.01</td><td>0.01</td><td>0.3</td><td>0.1</td><td>0.7</td><td>0.1</td><td>1.4</td></tr>
</table>

Table 2: Each proxy's runtime (in sec). N: number of proxies, t: threshold.

<table>
<tr><td>N, t</td><td>2, 1</td><td>4, 1</td><td>4, 3</td><td>8, 1</td><td>8, 7</td><td>16, 1</td><td>16, 15</td></tr>
<tr><td>KGenDeleg</td><td>0.7</td><td colspan="2">3.2</td><td colspan="2">14.4</td><td colspan="2">54.3</td></tr>
<tr><td>DerivePK</td><td>0.3</td><td>0.6</td><td>0.6</td><td>1.4</td><td>1.4</td><td>2.4</td><td>2.8</td></tr>
</table>

Table 3: Delegator's runtime (in sec). N: number of proxies, t: threshold.

Thank You for Your Attention!

Eibsee