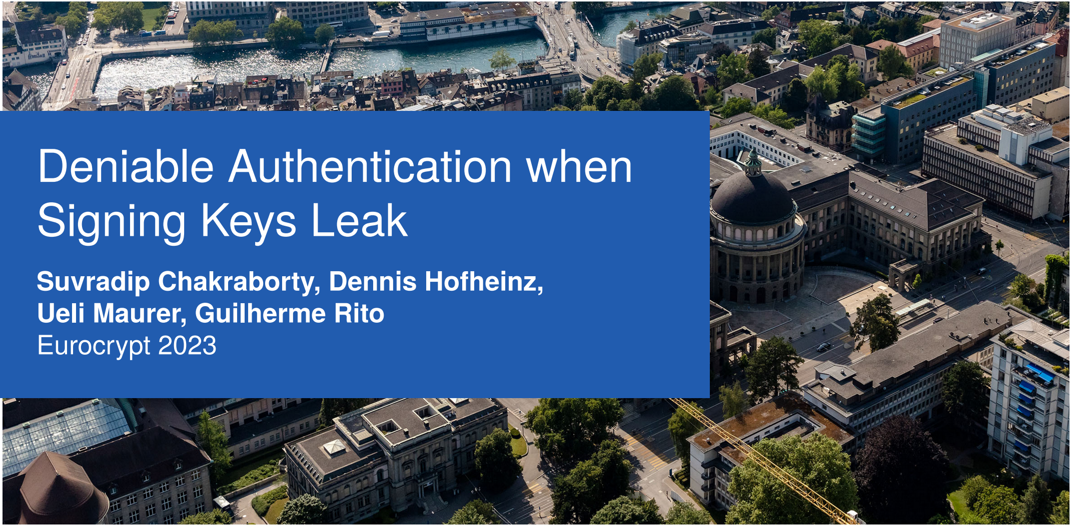
ETH zürich



Deniable Authentication when Signing Keys Leak

Suvradip Chakraborty, Dennis Hofheinz, Ueli Maurer, Guilherme Rito

Eurocrypt 2023

Outline

1. Motivation

Basic guarantees messaging apps should give:

1) Confidentiality

If a message is sent to *Bob*, only *Bob* can read it;

2) Authenticity

If *Bob* reads a message as coming from *Alice*, then *Alice* sent this message.

1. Motivation

Basic guarantees messaging apps should give:

1) Confidentiality

If a message is sent to *Bob*, only *Bob* can read it;

2) Authenticity

If *Bob* reads a message as coming from *Alice*, then *Alice* sent this message.

Long-lived sessions $\Rightarrow$ Increased chance of state leakage;

Messaging apps also aim to guarantee:

Forward Secrecy

If all parties want to delete the chat history, they can;

Backward Secrecy

Confidentiality can be recovered even after a party's state is leaked.

1. Problem

What if some parties want to keep a chat history?
$\Rightarrow$ **Forward Secrecy** gives no guarantee.[1]

[1] See, e.g., `https://faq.whatsapp.com/673193694148537`.

1. Problem

What if some parties want to keep a chat history?
$\Rightarrow$ **Forward Secrecy** gives no guarantee.[1]

What if some parties are dishonest?
$\Rightarrow$ Neither **Forward Secrecy**[1] nor **Backward Secrecy** give any guarantee.

[1] See, e.g., `https://faq.whatsapp.com/673193694148537`.

1. Problem

What if some parties want to keep a chat history?
⇒ **Forward Secrecy** gives no guarantee.[1]

What if some parties are dishonest?
⇒ Neither **Forward Secrecy**[1] nor **Backward Secrecy** give any guarantee.

What guarantees can we hope for in such cases?

[1] See, e.g., `https://faq.whatsapp.com/673193694148537`.

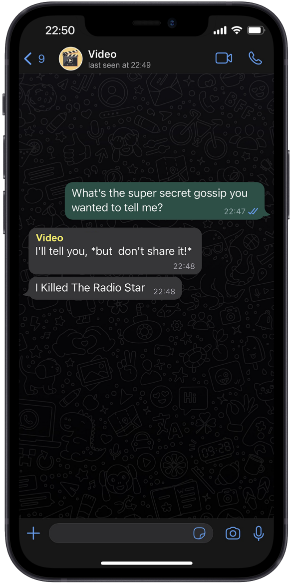
1. Guarantee: Off-The-Record Deniability

Parties in chat:

Buggles *dis*honest, the phone's owner;
Video honest, the killer.

Parties outside chat:

Judy *dis*honest, a judge.



1. Guarantee: Off-The-Record Deniability

Parties in chat:

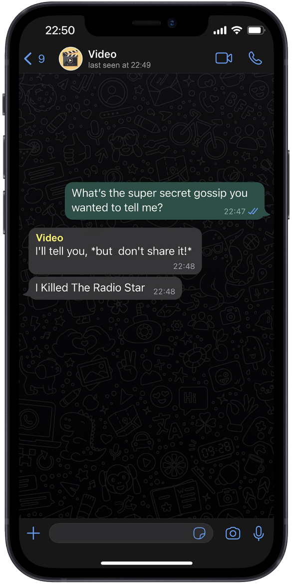
Buggles *dis*honest, the phone's owner;
Video honest, the killer.

Parties outside chat:

Judy *dis*honest, a judge.

Buggles' goal: convince *Judy* that
"Video Killed The Radio Star":

⇒ *Buggles* gives *Judy* its phone (with its secret keys).



1. Guarantee: Consistency

Video creates a (chat) group called *“Killer Confession”*;

Parties in the group:

Buggles honest;
Video *dis*honest;
Grace honest, an inspector looking for Radio Star’s killer.

1. Guarantee: Consistency

Video creates a (chat) group called *"Killer Confession"*;

Parties in the group:

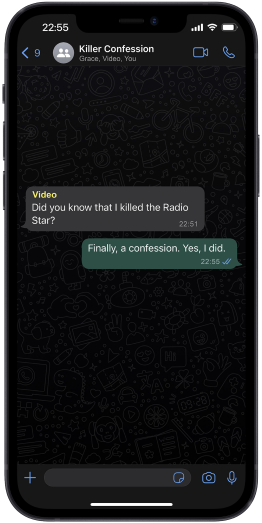
Buggles	honest;
Video	*dis*honest;
Grace	honest, an inspector looking for Radio Star's killer.

Video's goal: frame *Buggles* by falsifying a confession:

$\Rightarrow$ *Video* sends malformed ciphertexts that decrypt differently in *Buggles*' and *Grace*'s phones.

1. Guarantee: Consistency

Parties in the group: *Video* (dishonest), *Buggles* (honest, left), *Grace* (honest, right).



1. MDRS-PKE Schemes

Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE) Schemes, introduced in [2], guarantee:

1) Confidentiality;

2) Authenticity;

[2] Maurer et al. (Eurocrypt '22).

1. MDRS-PKE Schemes

Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE) Schemes, introduced in [2], guarantee:

1) Confidentiality;

2) Authenticity;

3) Off-The-Record;

4) Consistency;

[2] Maurer et al. (Eurocrypt '22).

1. MDRS-PKE Schemes

Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE) Schemes, introduced in [2], guarantee:

1) Confidentiality;

2) Authenticity;

3) Off-The-Record;

4) Consistency;

5) Anonymity.

[2] Maurer et al. (Eurocrypt '22).

1. Off-The-Record Deniability: A Closer Look

Parties in chat:

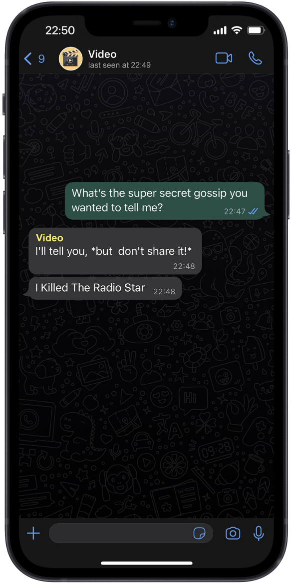
Buggles ***dis*****honest**, the phone's owner;
Video honest, the killer.

Parties outside chat:

Judy *dis*honest, a judge;

Buggles cooperates with *Judy* giving all its secret keys.

Judy should not be convinced that
"Video Killed The Radio Star";



1. Off-The-Record Deniability: A Closer Look

Parties in chat:

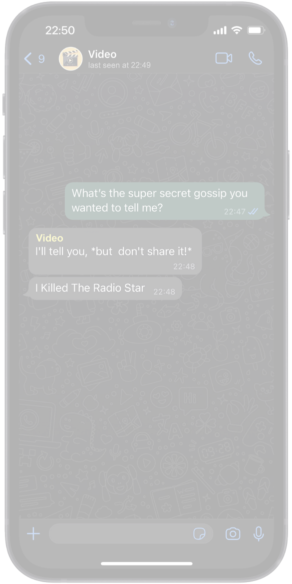
Buggles ~~dis~~**honest**, the phone's owner;
Video honest, the killer.

Parties outside chat:

Judy _dis_honest, a judge;
Eve **_dis_honest**, a third party.

Eve cooperates with *Judy* (i.e. tells on *Video* to *Judy*).

Judy should not be convinced that
"Video Killed The Radio Star";



1. Setting

Parties in chat: *Buggles* and *Video*.

Parties outside chat: *Judy* and *Eve*.

Honest *Buggles*:

Eve cooperates with *Judy* (i.e. tells on *Video* to *Judy*).

*Dis*honest *Buggles*:

Buggles cooperates with *Judy* giving all its secret keys.

1. Setting

Parties in chat: *Buggles* and *Video*.

Parties outside chat: *Judy* and *Eve*.

Honest *Buggles*: ***Judy* knows public information (only)**
Eve cooperates with *Judy* (i.e. tells on *Video* to *Judy*).

*Dis*honest *Buggles*: ***Judy* knows public information + *Buggles*' secret keys**
Buggles cooperates with *Judy* giving all its secret keys.

1. Setting

Parties in chat: *Buggles* and *Video*.

Parties outside chat: *Judy* and *Eve*.

Honest *Buggles*: ***Judy* knows public information (only)**
Eve cooperates with *Judy* (i.e. tells on *Video* to *Judy*).

*Dis*honest *Buggles*: ***Judy* knows public information + *Buggles*' secret keys**
Buggles cooperates with *Judy* giving all its secret keys.

Problem: What if *Judy* could get *Video*'s secret keys?
e.g. *Judy* could issue a warrant on *Video*'s secret keys.

1. Setting of Our Work

Parties in chat: *Buggles* and *Video*.

Parties outside chat: *Judy* and *Eve*.

Honest *Buggles*: ***Judy* knows public information (only) + *Video*'s secret keys**

Eve cooperates with *Judy* (i.e. tells on *Video* to *Judy*).

*Dis*honest *Buggles*: ***Judy* knows public information + *Buggles*' + *Video*'s secret keys**

Buggles cooperates with *Judy* giving all its secret keys.

Problem: What if *Judy* could get *Video*'s secret keys?

e.g. *Judy* could issue a warrant on *Video*'s secret keys.

1. Scalability of MDRS-PKE Schemes

[1] https://www.businessofapps.com/data/messaging-app-market/.

1. Scalability of MDRS-PKE Schemes — Types of scalability:

1) “Security” should be independent of #users and #messages;

$\Rightarrow$ Particularly important for Secure Messaging:

over 3 billion ($> 2^{31}$) active users[1];
over 100 billion ($> 2^{36}$) daily messages sent[1].

[1] https://www.businessofapps.com/data/messaging-app-market/.

1. Scalability of MDRS-PKE Schemes — Types of scalability:

1) “Security” should be independent of #users and #messages;

$\Rightarrow$ Particularly important for Secure Messaging:

over 3 billion ($> 2^{31}$) active users[1];
over 100 billion ($> 2^{36}$) daily messages sent[1].

2) For a group chat with n participants:

Ciphertext size, Encryption time, Decryption time } grow at most linearly with n.

[1] https://www.businessofapps.com/data/messaging-app-market/.

1. Scalability in Secure Messaging — Prior Work

In the existing MDRS-PKE construction:

“Security” degrades with #users;

For a group chat with n participants:

$$\left.\begin{array}{l}\text{Ciphertext size}\\ \text{Encryption time}\\ \text{Decryption time}\end{array}\right\}\text{ grow quadratically with } n.$$

1. Scalability in Secure Messaging — Prior Work ⇒ **Our Work**

In the existing MDRS-PKE construction:
First scalable MDRS-PKE construction:

- "Security" degrades with #users;[1]
- **"Security" independent of #users and #messages;**

For a group chat with n participants:

$$\left.\begin{array}{l}\text{Ciphertext size}\\ \text{Encryption time}\\ \text{Decryption time}\end{array}\right\}\text{ grow } \underline{\textbf{linearly}} \textbf{ with } n.$$

Outline

1. Introduction
2. MDRS-PKE Schemes
3. Main Idea of (M)DVS Construction

2. Constructing an MDRS-PKE Scheme

Maurer et al. give a very simple construction [2]:

Building blocks:

- Public Key Encryption for Broadcast Scheme (PKEBC);
- Multi-Designated Verifier Signature Scheme (MDVS).

[2] Maurer et al. (Eurocrypt '22).

2. Constructing an MDRS-PKE Scheme

Maurer et al. give a very simple construction [2]:

Building blocks:

- Public Key Encryption for Broadcast Scheme (PKEBC);
- Multi-Designated Verifier Signature Scheme (MDVS).

Main idea: Sign-then-Encrypt

- MDVS-sign(receivers' PKEBC public keys and message);
- PKEBC-encrypt(sender's + receivers' MDVS public keys, message and signature).

[2] Maurer et al. (Eurocrypt '22).

2. Scalability of the Scheme — Security

Security reductions to PKEBC and MDVS are tight:

Exists PKEBC with tight reductions to std. assumptions:

Exists MDVS with tight reductions to std. assumptions:

2. Scalability of the Scheme — Security

Security reductions to PKEBC and MDVS are tight: [2] ✓

Exists PKEBC with tight reductions to std. assumptions:

Exists MDVS with tight reductions to std. assumptions:

2. Scalability of the Scheme — Security

Security reductions to PKEBC and MDVS are tight: [2] ✓

Exists PKEBC with tight reductions to std. assumptions: [2] ✓

Exists MDVS with tight reductions to std. assumptions:

2. Scalability of the Scheme — Security

Security reductions to PKEBC and MDVS are tight: [2] ✓

Exists PKEBC with tight reductions to std. assumptions: [2] ✓

Exists MDVS with tight reductions to std. assumptions: ✗

2. Scalability of the Scheme — Efficiency

Consider a message sent to n receivers:

[1] Damgård et al. (TCC ’20).

2. Scalability of the Scheme — Efficiency

Consider a message sent to n receivers:

MDVS scheme from [1]:

$$\left.\begin{array}{l}\text{Signature size}\\ \text{Signing time}\\ \text{Verifying time}\end{array}\right\} \approx O(n) \checkmark$$

[1] Damgård et al. (TCC ’20).

2. Scalability of the Scheme — Efficiency

Consider a message sent to n receivers:

MDVS scheme from [1]:

$$\left.\begin{array}{l}\text{Signature size}\\\text{Signing time}\\\text{Verifying time}\end{array}\right\}\approx O(n)\ \checkmark$$

PKEBC scheme from [2]:

$$\left.\begin{array}{l}\text{Ciphertext size}\\\text{Encryption time}\\\text{Decryption time}\end{array}\right\}\approx O(n^2)\ \text{✗}$$

[1] Damgård et al. (TCC ’20).

2. Our Work

First MDVS scheme with tight reductions to std. assumptions:

Construction is conceptually simple;

$$\left.\begin{array}{l}\text{Signature size}\\\text{Signing time}\\\text{Verifying time}\end{array}\right\} \approx O(n).$$

2. Our Work

First MDVS scheme with tight reductions to std. assumptions:

Construction is conceptually simple;

$$\left.\begin{array}{l}\text{Signature size}\\ \text{Signing time}\\ \text{Verifying time}\end{array}\right\} \approx O(n).$$

First efficient PKEBC scheme:

$$\left.\begin{array}{l}\text{Ciphertext size}\\ \text{Encryption time}\\ \text{Decryption time}\end{array}\right\} \approx O(n).$$

2. Our Work

First MDVS scheme with tight reductions to std. assumptions:

Construction is conceptually simple;

$$\left.\begin{array}{l}\text{Signature size}\\ \text{Signing time}\\ \text{Verifying time}\end{array}\right\} \approx O(n).$$

First efficient PKEBC scheme:

$$\left.\begin{array}{l}\text{Ciphertext size}\\ \text{Encryption time}\\ \text{Decryption time}\end{array}\right\} \approx O(n).$$

Both schemes are proven tightly secure under adaptive corruptions.

2. Remainder of Presentation

Overview of tightly secure (Single Verifier) DVS scheme construction.

Outline

3. DVS Security Notions (Informal)

Authenticity:

Off-The-Record:

3. DVS Security Notions (Informal)

Authenticity: Adv. cannot forge signatures on unsigned messages.
(Adv. gets no secret key.)

Off-The-Record:

3. DVS Security Notions (Informal)

Authenticity: Adv. cannot forge signatures on unsigned messages.
(Adv. gets no secret key.)

Off-The-Record:

Dishonest verifier:

Honest verifier:

3. DVS Security Notions (Informal)

Authenticity: Adv. cannot forge signatures on unsigned messages.
(Adv. gets no secret key.)

Off-The-Record:

Dishonest verifier: Real sigs. $\approx_c$ dishonest verifier forgeries.
(Adv. gets signer's + verifier's secret keys.)

Honest verifier:

3. DVS Security Notions (Informal)

Authenticity: Adv. cannot forge signatures on unsigned messages.
(Adv. gets no secret key.)

Off-The-Record:

Dishonest verifier: Real sigs. $\approx_c$ dishonest verifier forgeries.
(Adv. gets signer's + verifier's secret keys.)

Honest verifier: Real sigs. $\approx_c$ publicly computable forgeries.
(Adv. gets signer's secret key.)

3. DVS Construction: First Attempt

Building blocks:

- Non Interactive Key Exchange (NIKE) scheme;
- Public Key Encryption (PKE) scheme.

3. DVS Construction: First Attempt

Building blocks:

- Non Interactive Key Exchange (NIKE) scheme;
- Public Key Encryption (PKE) scheme.

Signer key-pair: NIKE key-pair.

Verifier key-pair: NIKE key-pair + PKE key-pair.

3. DVS Construction: First Attempt

Building blocks:
- Non Interactive Key Exchange (NIKE) scheme;
- Public Key Encryption (PKE) scheme.

Signer key-pair: NIKE key-pair.

Verifier key-pair: NIKE key-pair + PKE key-pair.

Sign$(\mathtt{sk}_{\text{Signer}} := \mathtt{sk}_{\text{NIKE}}, \mathtt{pk}_{\text{Verifier}} := (\mathtt{pk}_{\text{NIKE}}, \mathtt{pk}_{\text{PKE}}), m)$:
- Compute NIKE shared key ($\mathtt{k}_{\mathtt{shared}}$) of sender and verifier;
- Encrypt $(\mathtt{k}_{\mathtt{shared}} \mid\mid m)$ under $\mathtt{pk}_{\text{PKE}}$; output the ciphertext.

3. DVS Construction: First Attempt

Building blocks:

Non Interactive Key Exchange (NIKE) scheme;

Public Key Encryption (PKE) scheme.

Signer key-pair: NIKE key-pair.

Verifier key-pair: NIKE key-pair + PKE key-pair.

Sign$(\mathtt{sk}_{\text{Signer}} := \mathtt{sk}_{\text{NIKE}}, \mathtt{pk}_{\text{Verifier}} := (\mathtt{pk}_{\text{NIKE}}, \mathtt{pk}_{\text{PKE}}), m)$:

Compute NIKE shared key ($\mathtt{k}_{\mathtt{shared}}$) of sender and verifier;

Encrypt $(\mathtt{k}_{\mathtt{shared}} \,||\, m)$ under $\mathtt{pk}_{\text{PKE}}$; output the ciphertext.

Intuitively, guarantees:

Authenticity: ✓

Off-The-Record: ✓ (Even when $\mathtt{sk}_{\text{Signer}}$ leaks to adv.)

3. DVS Construction: First Attempt

Building blocks:

- Non Interactive Key Exchange (NIKE) scheme;
- Public Key Encryption (PKE) scheme.

Signer key-pair: NIKE key-pair.

Verifier key-pair: NIKE key-pair + PKE key-pair.

Sign$(\mathtt{sk}_{\text{Signer}} := \mathtt{sk}_{\text{NIKE}}, \mathtt{pk}_{\text{Verifier}} := (\mathtt{pk}_{\text{NIKE}}, \mathtt{pk}_{\text{PKE}}), m)$:

- Compute NIKE shared key ($\mathtt{k}_{\mathtt{shared}}$) of sender and verifier;
- Encrypt $(\mathtt{k}_{\mathtt{shared}} \,||\, m)$ under $\mathtt{pk}_{\text{PKE}}$; output the ciphertext.

Intuitively, guarantees:

- **Authenticity:** ✓
- **Off-The-Record:** ✓ (Even when $\mathtt{sk}_{\text{Signer}}$ leaks to adv.)

Problem: Unknown if tightly secure NIKE schemes exist (under std. assumptions).

3. DVS Construction: What is the role of the NIKE?

3. DVS Construction: What is the role of the NIKE?

Authenticity: $\mathtt{k}_{\mathtt{shared}}$ only computable with $\mathtt{sk}_{\mathsf{Signer}}$ or $\mathtt{sk}_{\mathsf{Verifier}}$.

Off-The-Record: Dishonest verifier can forge sigs. by computing $\mathtt{k}_{\mathtt{shared}}$.

3. DVS Construction: What is the role of the NIKE?

Authenticity: $\mathtt{k}_{\mathtt{shared}}$ only computable with $\mathtt{sk}_{\text{Signer}}$ or $\mathtt{sk}_{\text{Verifier}}$.

Off-The-Record: Dishonest verifier can forge sigs. by computing $\mathtt{k}_{\mathtt{shared}}$.

NIKE is used to prove an OR-statement:

"The signer really signed m ***OR*** the verifier forged a signature on m."

3. DVS Construction

Building blocks:

Non Interactive Zero Knowledge (NIZK) scheme;
PKE scheme;
One Way Function (OWF) F.

Public parameters:

$\texttt{pp} := (\texttt{pk}_{\texttt{pp}}, \texttt{crs})$, where $\texttt{pk}_{\texttt{pp}}$ is a PKE public key.

Signer key-pair:

$(\texttt{pk}_{\text{Signer}}, \texttt{sk}_{\text{Signer}}) := (y_s = F(x_s), x_s)$, where x_s is a OWF pre-image.

Verifier key-pair:

$\texttt{pk}_{\text{Verifier}} := (y_v = F(x_v), \texttt{pk}_{\text{PKE}})$;
$\texttt{sk}_{\text{Verifier}} := (x_v, \texttt{sk}_{\text{PKE}})$.

3. DVS Construction

Sign$(\mathtt{pk}_{\mathtt{pp}}, \mathtt{sk}_{\mathsf{Signer}} := \underline{\boldsymbol{x_s}}, \mathtt{pk}_{\mathsf{Verifier}} := (y_v, \mathtt{pk}_{\mathsf{PKE}}), m)$:

$c_{\mathrm{pp}} \leftarrow \mathsf{Enc}(\mathtt{pk}_{\mathtt{pp}}, (m, \underline{\boldsymbol{x_s}}, 1))$;

$c \leftarrow \mathsf{Enc}(\mathtt{pk}_{\mathsf{verifier}}, 1)$;

$\pi \leftarrow$ NIZK-Prove(statement $s :=$

"c encrypts $1 \Rightarrow c_{\mathrm{pp}}$ encrypts a pre-image of either $\underline{\boldsymbol{y_s}}$ or $\underline{\boldsymbol{y_v}}$, and m");

$\sigma := (c_{\mathrm{pp}}, c, \pi)$.

3. DVS Construction

Sign($\mathtt{pk}_{\mathrm{pp}}, \mathtt{sk}_{\mathsf{Signer}} := \underline{\boldsymbol{x_s}}, \mathtt{pk}_{\mathsf{Verifier}} := (y_v, \mathtt{pk}_{\mathsf{PKE}}), m$):
$c_{\mathrm{pp}} \leftarrow$ Enc($\mathtt{pk}_{\mathrm{pp}}, (m, \underline{\boldsymbol{x_s}}, 1)$);
$c \leftarrow$ Enc($\mathtt{pk}_{\mathsf{verifier}}, 1$);
$\pi \leftarrow$ NIZK-Prove(statement $s :=$
"c encrypts $1 \Rightarrow c_{\mathrm{pp}}$ encrypts a pre-image of either $\underline{\boldsymbol{y_s}}$ or $\underline{\boldsymbol{y_v}}$, and m");
$\sigma := (c_{\mathrm{pp}}, c, \pi)$.

Vfy($\mathtt{pk}_{\mathrm{pp}}, \mathtt{pk}_{\mathsf{Signer}} := y_s, \mathtt{sk}_{\mathsf{Verifier}} := (x_v, \mathtt{sk}_{\mathsf{PKE}}), m, \sigma := (c_{\mathrm{pp}}, c, \pi)$):
well-formed $\leftarrow$ NIZK-Verify(π, statement s);
$b \leftarrow$ PKE.$D_{\mathtt{sk}_{\mathsf{PKE}}}(c)$;
Output ($b \wedge$ well-formed).

3. DVS Construction

$\mathsf{Forge}_{\mathsf{Verifier}}(\mathtt{pk}_{\mathtt{pp}}, \mathtt{pk}_{\mathsf{Signer}} := y_s, \mathtt{sk}_{\mathsf{Verifier}} := (\underline{\boldsymbol{x_v}}, \mathtt{sk}_{\mathsf{PKE}}), m)$:

$c_{\mathrm{pp}} = \mathsf{Enc}(\mathtt{pk}_{\mathtt{pp}}, (m, \underline{\boldsymbol{x_v}}, 1))$;

$c = \mathsf{Enc}(\mathtt{pk}_{\mathsf{verifier}}, 1)$;

$\pi = \text{NIZK-Prove(statement } s)$;

$\sigma = (c_{\mathrm{pp}}, c, \pi)$.

3. DVS Construction

$\text{Forge}_{\text{Verifier}}(\mathtt{pk}_{\mathtt{pp}}, \mathtt{pk}_{\text{Signer}} := y_s, \mathtt{sk}_{\text{Verifier}} := (\underline{\boldsymbol{x_v}}, \mathtt{sk}_{\text{PKE}}), m)$:

$c_{\text{pp}} = \text{Enc}(\mathtt{pk}_{\mathtt{pp}}, (m, \underline{\boldsymbol{x_v}}, 1))$;
$c = \text{Enc}(\mathtt{pk}_{\text{verifier}}, 1)$;
$\pi = \text{NIZK-Prove(statement } s)$;
$\sigma = (c_{\text{pp}}, c, \pi)$.

$\text{Forge}_{\text{Public}}(\mathtt{pk}_{\mathtt{pp}}, \mathtt{pk}_{\text{Signer}} := y_s, \mathtt{pk}_{\text{Verifier}} := (y_v, \mathtt{pk}_{\text{PKE}}), m)$:

$c_{\text{pp}} = \text{Enc}(\mathtt{pk}_{\mathtt{pp}}, (m, \underline{\mathbf{0}}, 0))$;
$c = \text{Enc}(\mathtt{pk}_{\text{verifier}}, \underline{\mathbf{0}})$;
$\pi = \text{NIZK-Prove(statement } s)$;
$\sigma = (c_{\text{pp}}, c, \pi)$.

Thank you!

Bibliography

[1] Ivan Damgård, Helene Haagh, Rebekah Mercer, Anca Nitulescu, Claudio Orlandi, and Sophia Yakoubov.
Stronger security and constructions of multi-designated verifier signatures.
In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 229–260. Springer, Heidelberg, November 2020.

[2] Ueli Maurer, Christopher Portmann, and Guilherme Rito.
Multi-designated receiver signed public key encryption.
In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 644–673. Springer, Heidelberg, May / June 2022.