Cryptography from Zero Knowledge
Advanced Security and New Constructions

PhD Defense

Akira Takahashi

AARHUS UNIVERSITET

Victor

$$x \overset{?}{\in} L$$

$x \overset{?}{\in} L$

?

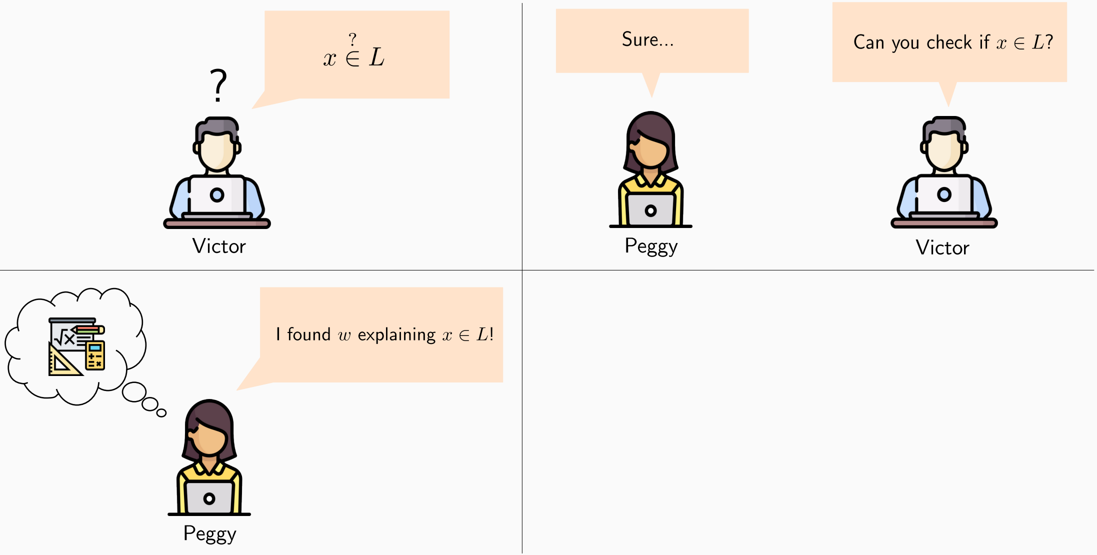
Victor

Sure...

Peggy

Can you check if $x \in L$?

Victor



$x \overset{?}{\in} L$

?

Victor

Sure...

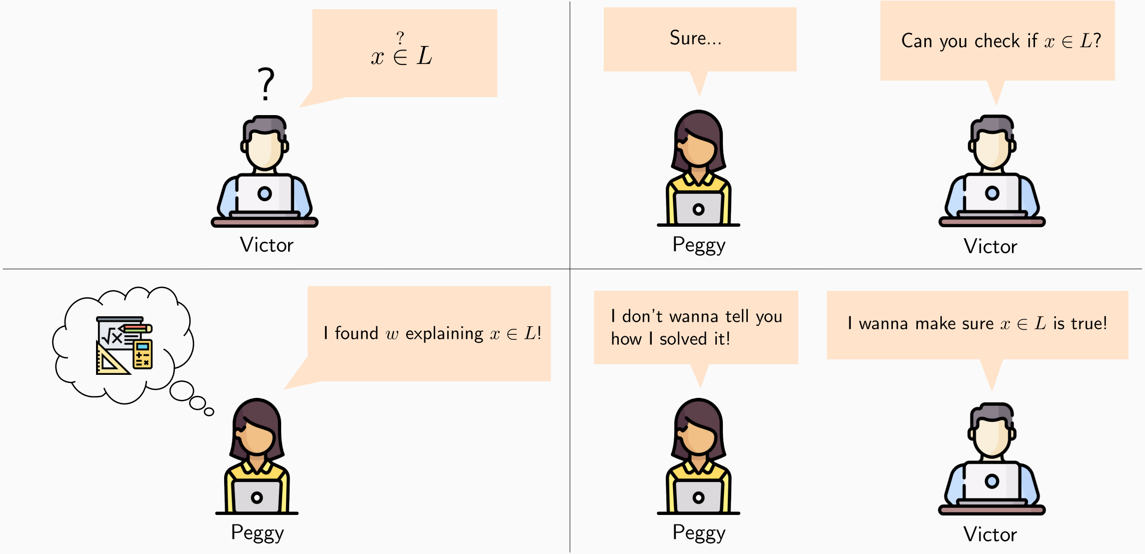
Peggy

Can you check if $x \in L$?

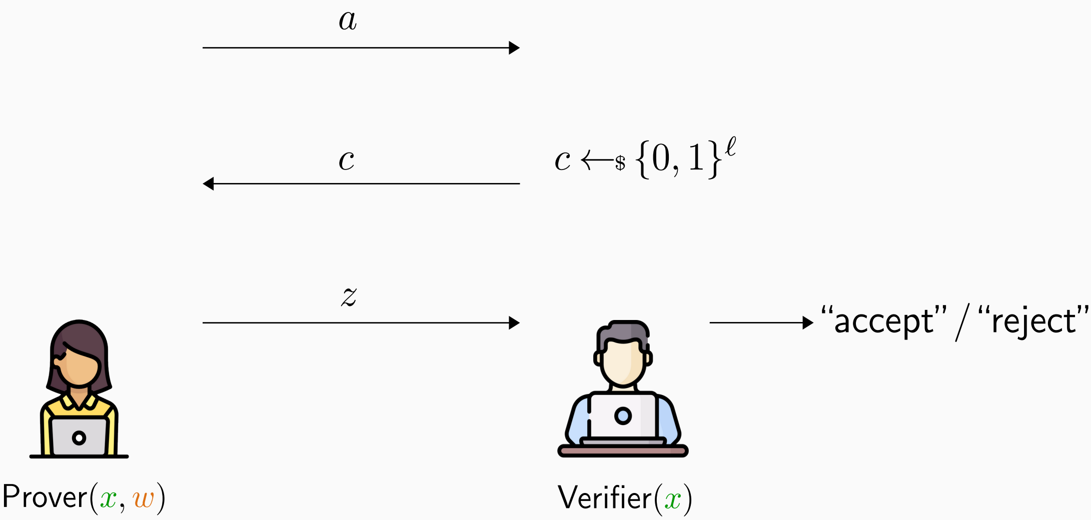
Victor

I found w explaining $x \in L$!

Peggy


?
$x \overset{?}{\in} L$
Victor
Sure...
Peggy
Can you check if $x \in L$?
Victor
I found w explaining $x \in L$!
Peggy
I don't wanna tell you how I solved it!
Peggy
I wanna make sure $x \in L$ is true!
Victor

Public Coin Interactive Proof



a

c

$c \leftarrow_{\$} \{0,1\}^{\ell}$

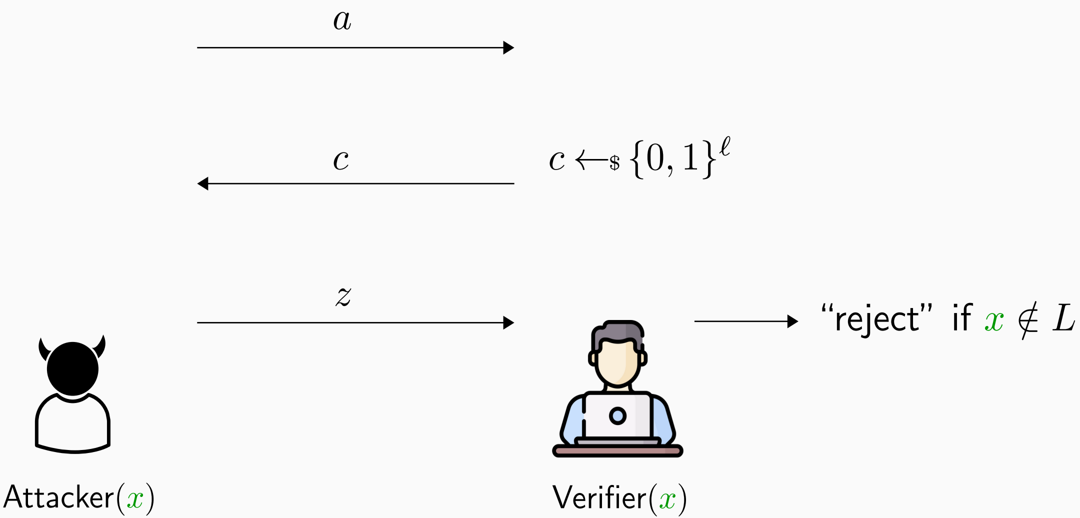
z

"accept" / "reject"

Prover(x, w)

Verifier(x)

Soundness



a

c $\quad c \leftarrow_{\$} \{0,1\}^{\ell}$

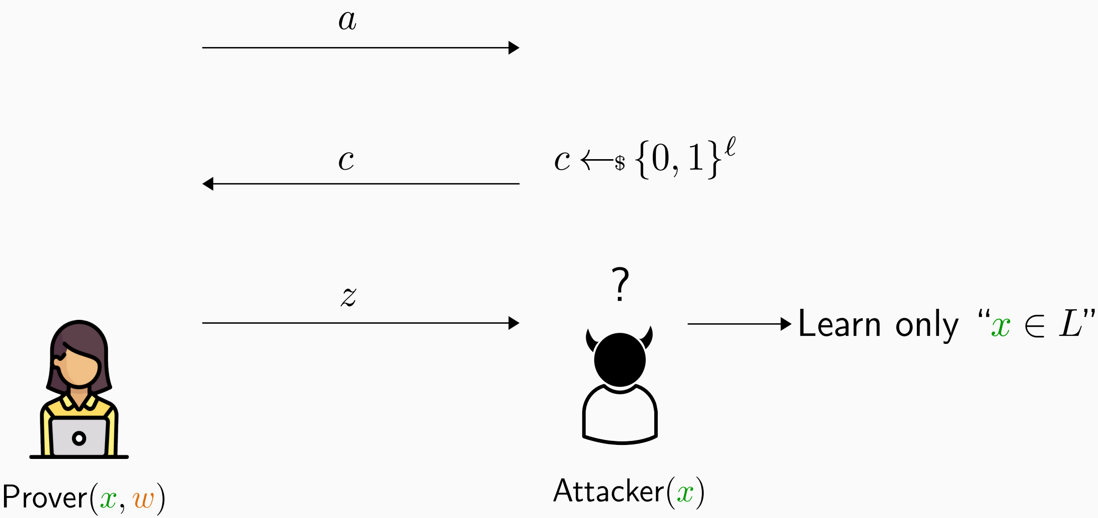
z

“reject” if $x \notin L$

Attacker(x)

Verifier(x)

Zero Knowledge



a

c

$c \leftarrow_{\$} \{0,1\}^{\ell}$

z

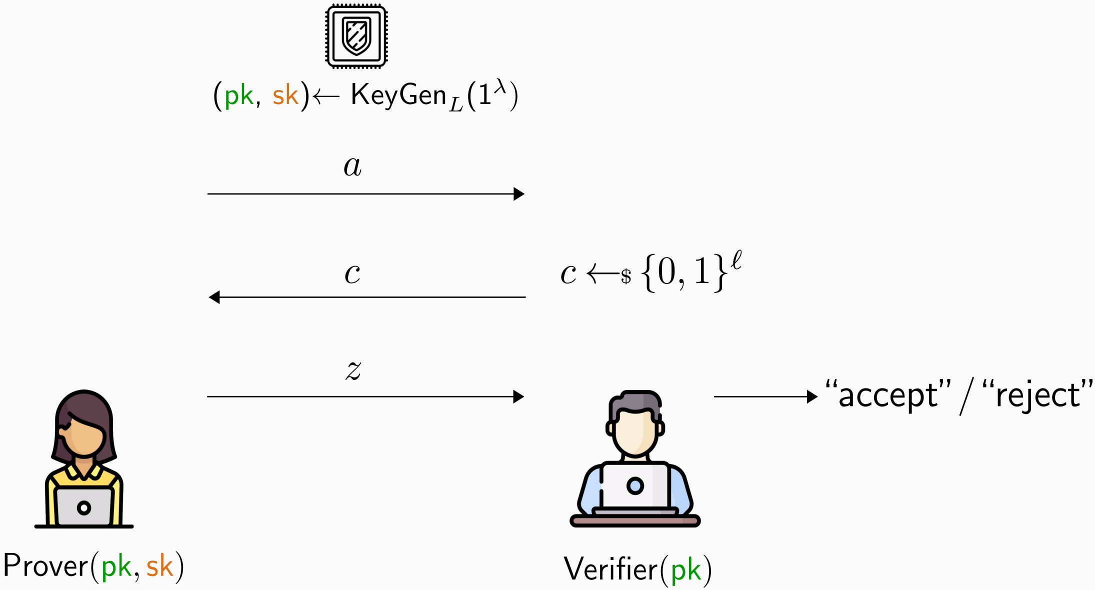
?

Learn only "$x \in L$"

Prover(x, w)

Attacker(x)

Identification from Interactive Proof



$(\textsf{pk}, \textsf{sk}) \leftarrow \textsf{KeyGen}_L(1^\lambda)$

a

c

$c \leftarrow_\$ \{0,1\}^\ell$

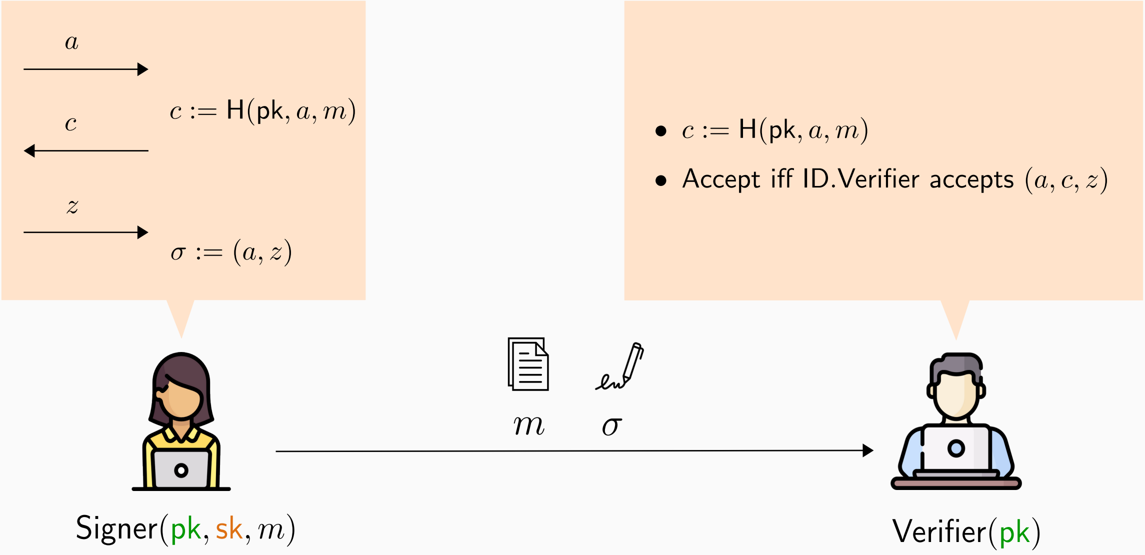
z

"accept" / "reject"

Prover(pk, sk)

Verifier(pk)

Signature from Identification: Fiat-Shamir Transform



$\xrightarrow{a}$

$\xleftarrow{c}$ $\quad c := \mathsf{H}(\mathsf{pk}, a, m)$

$\xrightarrow{z}$

$\sigma := (a, z)$

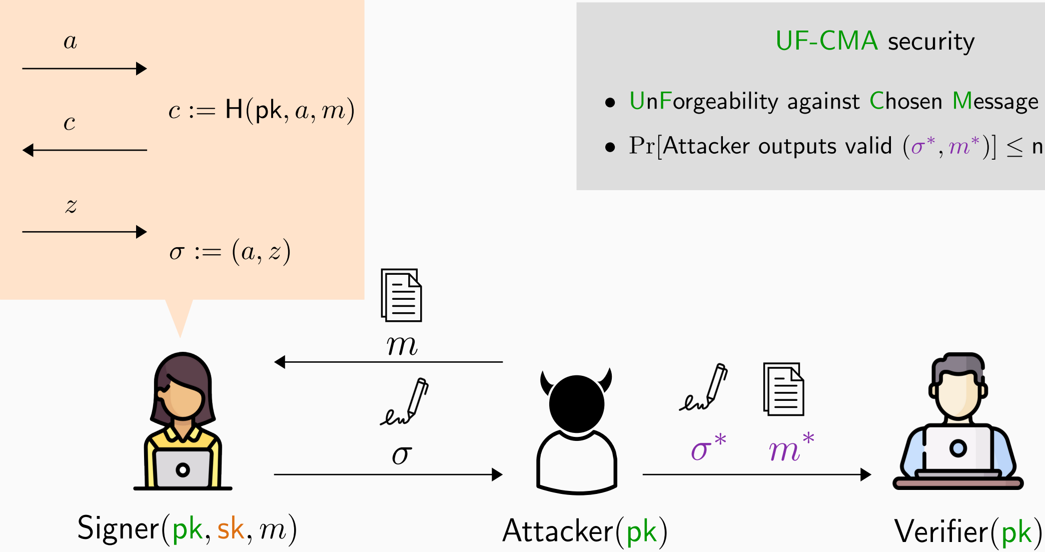
- $c := \mathsf{H}(\mathsf{pk}, a, m)$
- Accept iff ID.Verifier accepts (a, c, z)

$m \quad \sigma$

Signer(pk, sk, m)

Verifier(pk)

Security Notion for Digital Signatures



a

c

z

$c := \mathsf{H}(\mathsf{pk}, a, m)$

$\sigma := (a, z)$

m

σ

σ^* m^*

$\mathsf{Signer}(\mathsf{pk}, \mathsf{sk}, m)$

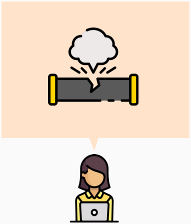
$\mathsf{Attacker}(\mathsf{pk})$

$\mathsf{Verifier}(\mathsf{pk})$

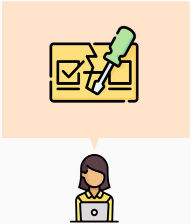
UF-CMA security

- UnForgeability against Chosen Message Attacks
- $\Pr[\text{Attacker outputs valid } (\sigma^*, m^*)] \leq \mathsf{negl}(\lambda)$

Advanced Security

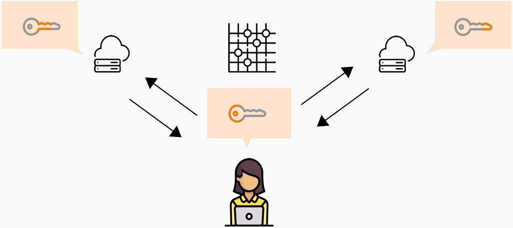


What happens if the signer partially leaks randomness?



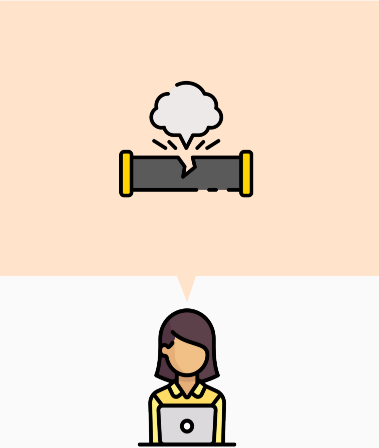
What happens if the signer produces faulty signatures?

New Constructions



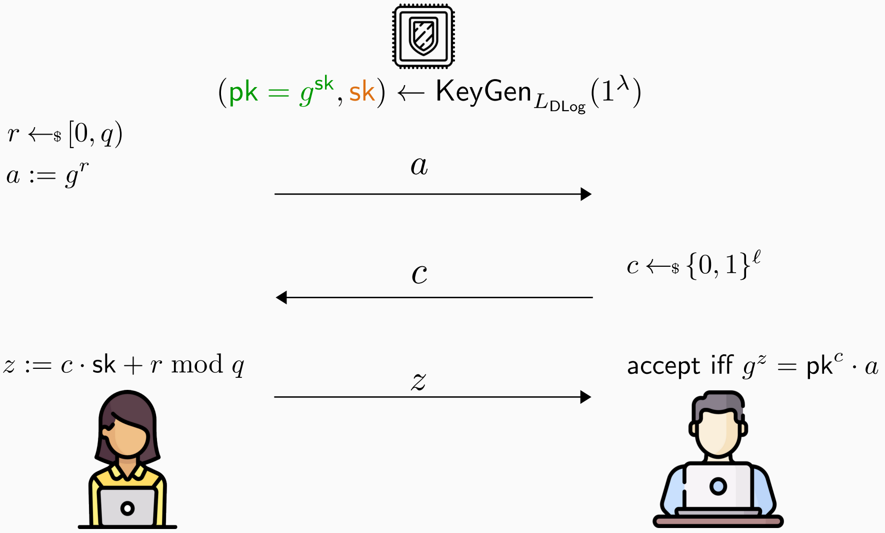
Can we construct multi-party signatures from lattice ZK proof?

Can we add verifiability to ciphertexts using ZK proof?



What happens if the signer partially leaks randomness?

Canonical Example: Schnorr Identification



$(\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_{L_{\mathsf{DLog}}}(1^{\lambda})$

$r \leftarrow_{\$} [0, q)$

$a := g^{r}$

$\xrightarrow{a}$

$c \leftarrow_{\$} \{0, 1\}^{\ell}$

$\xleftarrow{c}$

$z := c \cdot \mathsf{sk} + r \bmod q$

$\xrightarrow{z}$

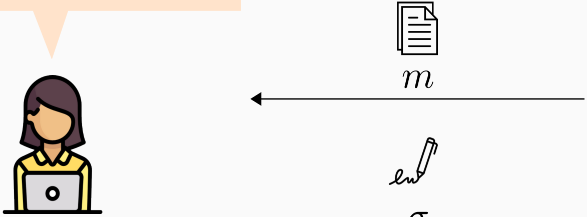
accept iff $g^{z} = \mathsf{pk}^{c} \cdot a$

Prover(pk, sk)

Verifier(pk)

Canonical Example: Schnorr Signature

```
$r \leftarrow_{\$} [0, q)$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```

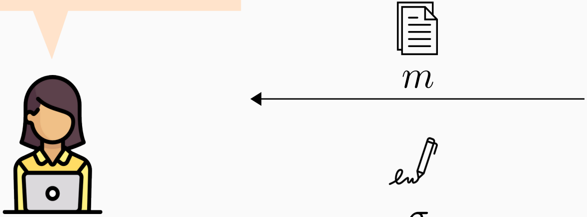


Signer($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m$)

Warm-up: What If Randomness is Reused?

Fixed r

```
$r := 111111\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```



m

σ

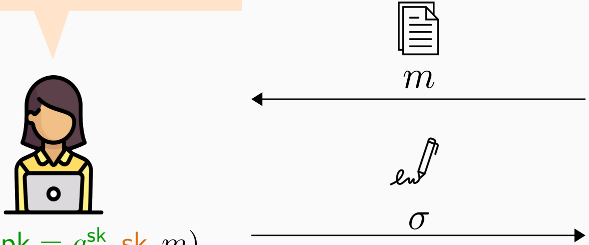
$\mathsf{Signer}(\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m)$

Warm-up: What If Randomness is Reused?

Fixed r

```
$r := 111111\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```

- Given (a, c, z) and (a, c', z'), $\mathsf{sk} = (z - z')(c - c')^{-1}$
- **NEVER** reuse r!!



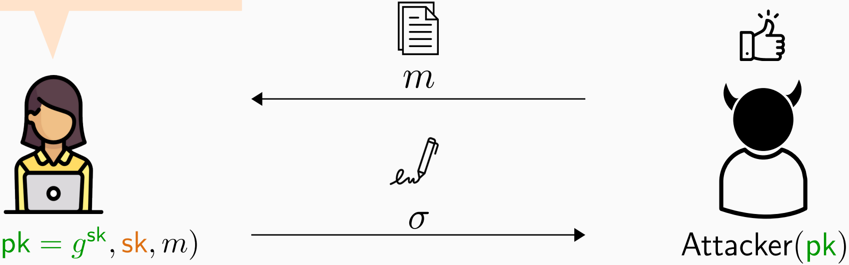
Signer($\mathsf{pk} = g^{\mathsf{sk}}$, sk, m)

Warm-up: What If Randomness is Reused?

Fixed r

```
$r := 111111\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```

- Given (a, c, z) and (a, c', z'), $\mathsf{sk} = (z - z')(c - c')^{-1}$
- **NEVER** reuse r!!



Signer($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m$)

Attacker(pk)

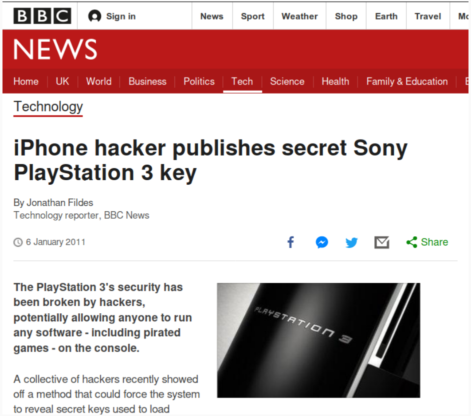
Randomness Failure in the Real World

- Poorly designed/implemented RNGs
- Predictable seed (`srand(time(0))`)
- **Side-channel attacks**:

  2018 CacheQuote on SGX EPID; PortSmash on SMT/Hyper-Threading; ROHNP

  2019 TPM-FAIL; Minerva; biased wolfSSL DSA

  2020 Déjà Vu attack on Mozilla's NSS; Raccoon attack on TLS 1.2



iPhone hacker publishes secret Sony PlayStation 3 key

By Jonathan Fildes
Technology reporter, BBC News

6 January 2011

The PlayStation 3's security has been broken by hackers, potentially allowing anyone to run any software - including pirated games - on the console.

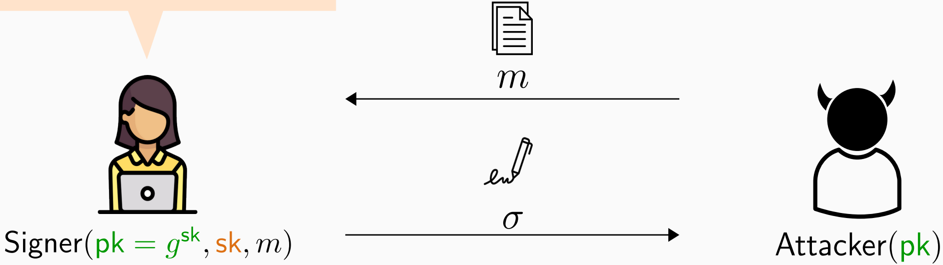
A collective of hackers recently showed off a method that could force the system to reveal secret keys used to load

BBC news. 2011. `https://www.bbc.com/news/technology-12116051`

Sensitivity of Randomness

Biased r

```
$r := 101101\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```



m

σ

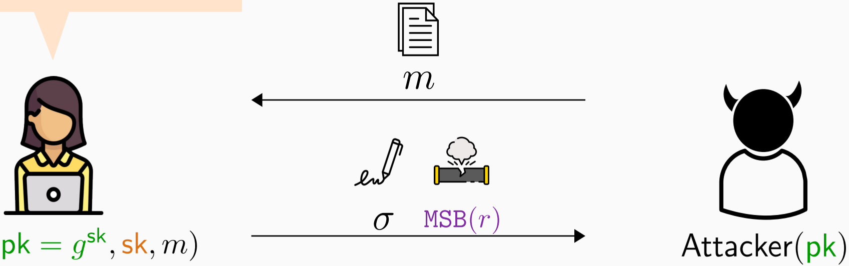
Signer($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m$)

Attacker(pk)

Sensitivity of Randomness

Leaky r

```
$r := 101101\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```



m

σ $\mathtt{MSB}(r)$

$\text{Signer}(\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m)$

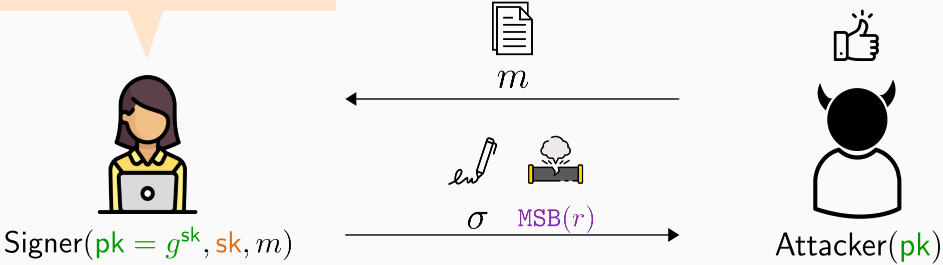
$\text{Attacker}(\mathsf{pk})$

Sensitivity of Randomness

Leaky r

```
$r := 101101\ldots$
$a := g^r$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := c \cdot \mathsf{sk} + r \bmod q$
$\sigma := (a, z)$
```

- sk can be still recovered by solving the **Hidden Number Problem**!



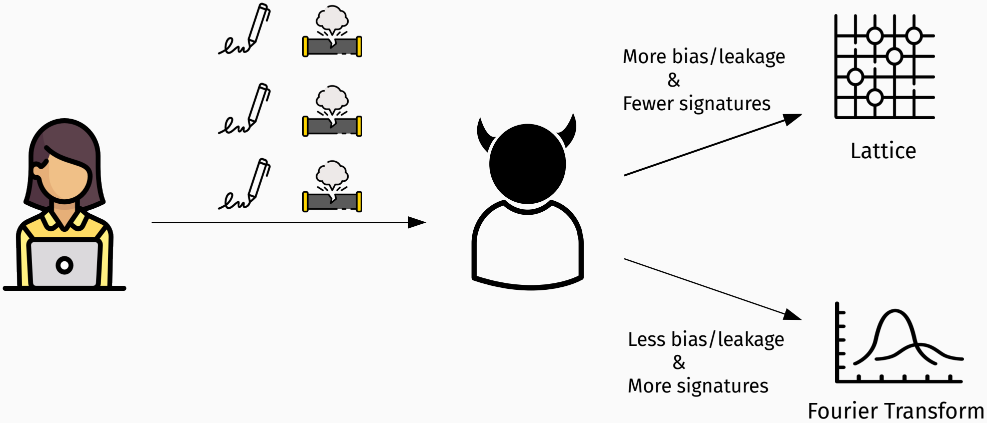
m

σ $\mathtt{MSB}(r)$

Signer($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}, m$)

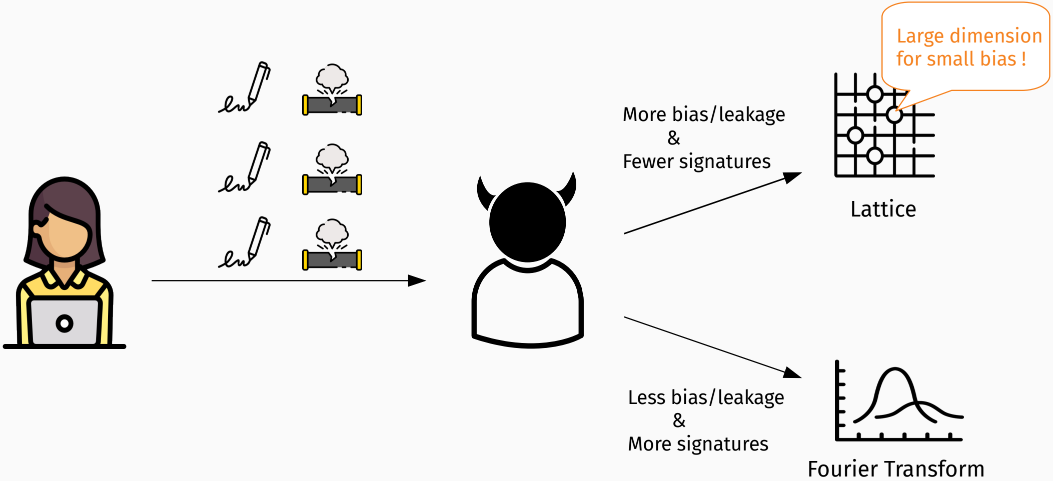
Attacker(pk)

How to Solve HNP



- Q. Can we reduce the number of signatures for the Fourier transform attack?
- Q. Can we attack even **less than 1-bit of leakage** per signature?
  - Attacker only learns correct $\mathsf{MSB}(r)$ with prob. < 1

How to Solve HNP



More bias/leakage
&
Fewer signatures

Large dimension for small bias !

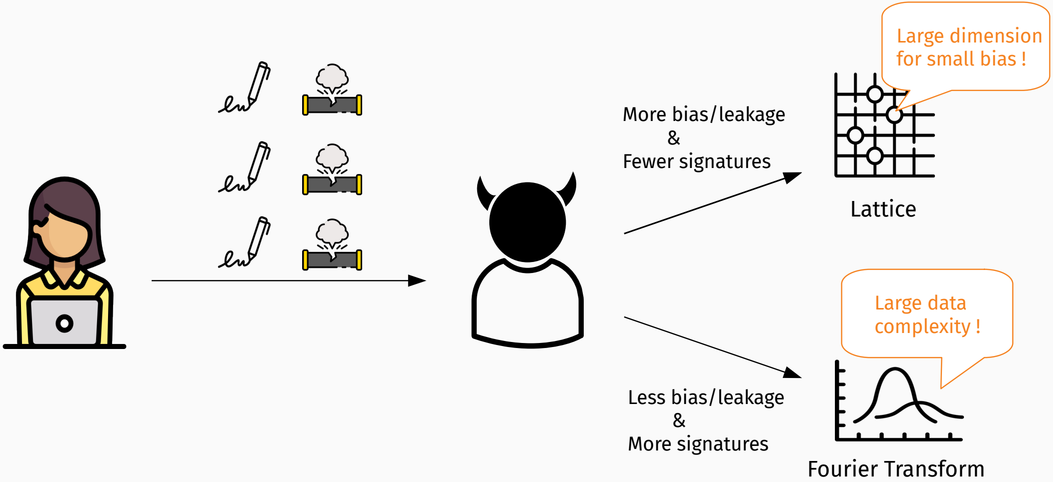
Lattice

Less bias/leakage
&
More signatures

Fourier Transform

- Q. Can we reduce the number of signatures for the Fourier transform attack?
- Q. Can we attack even **less than 1-bit of leakage** per signature?
  - Attacker only learns correct MSB(r) with prob. < 1

How to Solve HNP



More bias/leakage
&
Fewer signatures

Lattice

Large dimension for small bias !

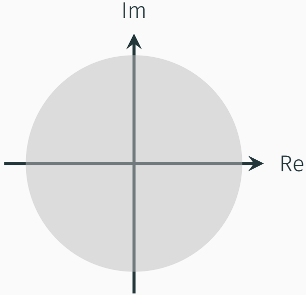
Less bias/leakage
&
More signatures

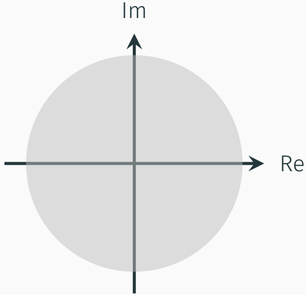
Fourier Transform

Large data complexity !

- Q. Can we reduce the number of signatures for the Fourier transform attack?
- Q. Can we attack even **less than 1-bit of leakage** per signature?
  - Attacker only learns correct MSB(r) with prob. < 1

Bleichenbacher’s Method: Quantifying Bias Using DFT



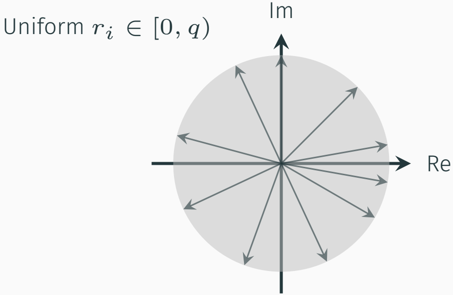


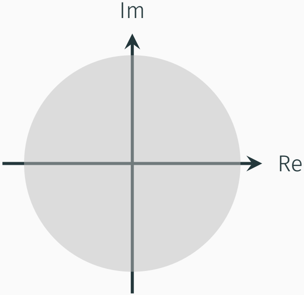
Definition

The **sampled bias** of points $K = (r_i)_{i\in\{1,\ldots,N\}}$ in $\mathbb{Z}_q$ is defined by

$$B_q(K) := \frac{1}{N}\sum_{i=1}^{N} e^{2\pi \mathrm{i} r_i/q}.$$

Bleichenbacher’s Method: Quantifying Bias Using DFT



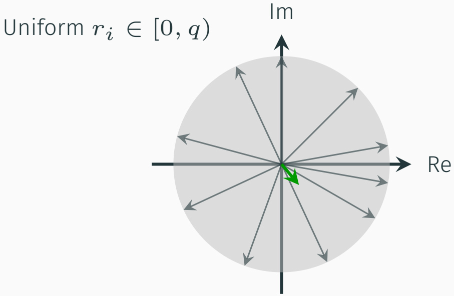


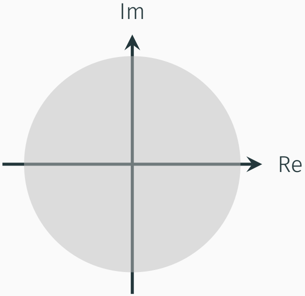
Definition

The **sampled bias** of points $K = (r_i)_{i\in\{1,\dots,N\}}$ in $\mathbb{Z}_q$ is defined by

$$B_q(K) := \frac{1}{N}\sum_{i=1}^{N} e^{2\pi \mathsf{i} r_i/q}.$$

Bleichenbacher’s Method: Quantifying Bias Using DFT



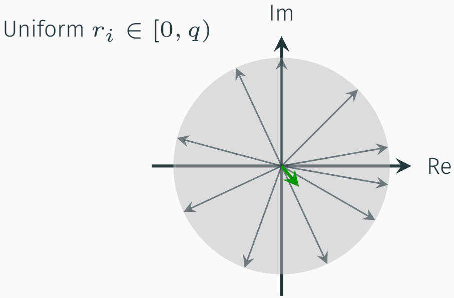


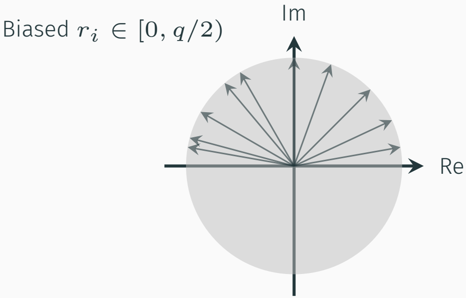
Definition

The **sampled bias** of points $K = (r_i)_{i \in \{1,\ldots,N\}}$ in $\mathbb{Z}_q$ is defined by

$$B_q(K) := \frac{1}{N} \sum_{i=1}^{N} e^{2\pi \mathrm{i} r_i / q}.$$

Bleichenbacher’s Method: Quantifying Bias Using DFT



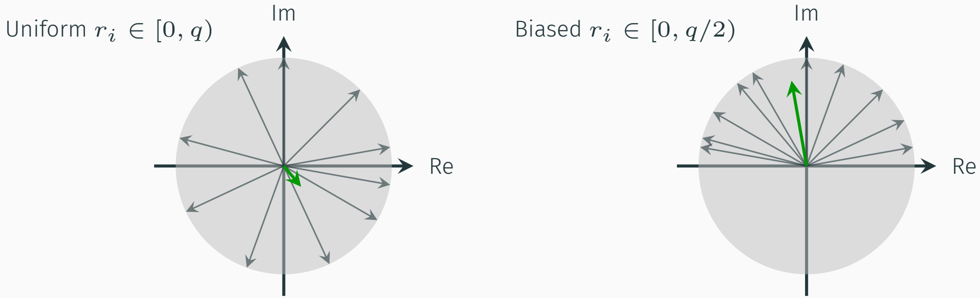


Definition

The **sampled bias** of points $K = (r_i)_{i\in\{1,\dots,N\}}$ in $\mathbb{Z}_q$ is defined by

$$B_q(K) := \frac{1}{N}\sum_{i=1}^{N} e^{2\pi \mathrm{i} r_i/q}.$$

Bleichenbacher’s Method: Quantifying Bias Using DFT


Uniform $r_i \in [0, q)$

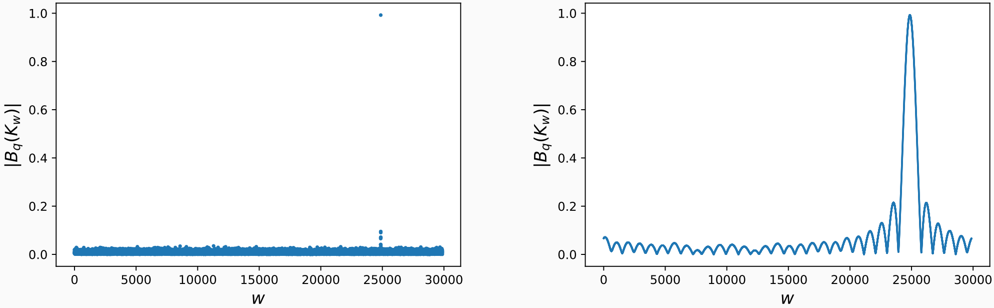
Biased $r_i \in [0, q/2)$

Definition

The **sampled bias** of points $K = (r_i)_{i\in\{1,\dots,N\}}$ in $\mathbb{Z}_q$ is defined by

$$B_q(K) := \frac{1}{N}\sum_{i=1}^{N} e^{2\pi \mathrm{i} r_i/q}.$$

Stretching the Peak Width

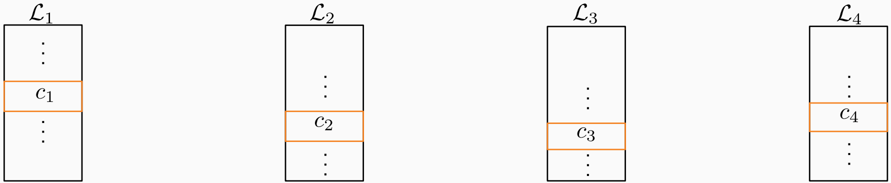


(a) $c_i \leq q$

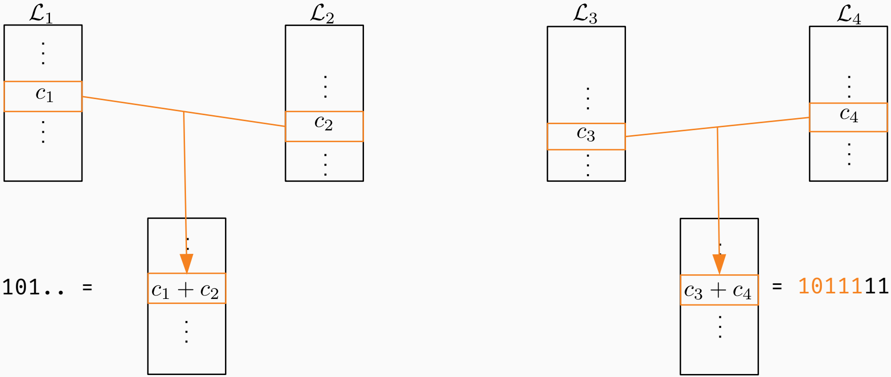
(b) $c_i' \ll q$

- $\mathbf{w}$: “guessed” secret key sk
- Naive way: find $\mathbf{w}$ that maximizes $|B_q((r_i = z_i - c_i \cdot \mathbf{w} \bmod q)_{i=1}^N)|$
- Crucial: construct $(c_i')_{i=1}^{N'}$ by taking **small** and **sparse** linear combinations of $(c_i)_{i=1}^N$

Our Approach: Generalized Birthday Problem



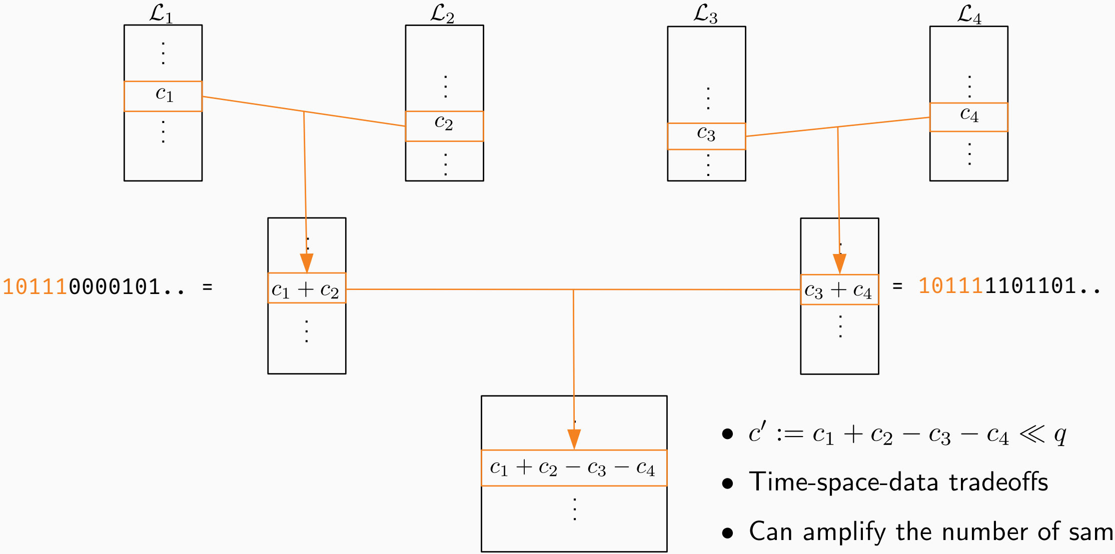
Our Approach: Generalized Birthday Problem



101110000101.. = $c_1 + c_2$

$c_3 + c_4$ = 101111101101..

Our Approach: Generalized Birthday Problem



- $c' := c_1 + c_2 - c_3 - c_4 \ll q$
- Time-space-data tradeoffs
- Can amplify the number of samples!

Experimental Records: Key Recovery Attack on ECDSA

Target	Bias	Facility	Error rate	Input	Thread	Time	RAM	Recovered
					(GBP)	(GBP)	(GBP)	MSBs
NIST P-192	1-bit	AWS EC2	0	2^{29}	2304	113h	492GB	39
NIST P-192	1-bit	AWS EC2	1%	2^{35}	2304	52h	492GB	39
`sect163r1`	1-bit	Cluster	0	2^{23}	256	7h	80GB	36
`sect163r1`	1-bit	Workstation	2.7%	2^{24}	48	42h	250GB	35
`sect163r1`	2-bit	Workstation	0	1024	16	2h	96GB	32

Table 1: Computational results for the first round of Bleichenbacher

- Attack on **P-192** is made possible by our highly optimized parallel implementation.
- Attack on `sect163r1` requires much less signatures

Experimental Records: Key Recovery Attack on ECDSA

Target	Bias	Facility	Error rate	Input	Thread (GBP)	Time (GBP)	RAM (GBP)	Recovered MSBs
NIST P-192	1-bit	AWS EC2	0	2^{29}	2304	113h	492GB	39
NIST P-192	1-bit	AWS EC2	1%	2^{35}	2304	52h	492GB	39
`sect163r1`	1-bit	Cluster	0	2^{23}	256	7h	80GB	36
`sect163r1`	1-bit	Workstation	2.7%	2^{24}	48	42h	250GB	35
`sect163r1`	2-bit	Workstation	0	1024	16	2h	96GB	32

Table 1: Computational results for the first round of Bleichenbacher

- Attack on **P-192** is made possible by our highly optimized parallel implementation.
- Attack on `sect163r1` requires much less signatures

Experimental Records: Key Recovery Attack on ECDSA

Target	Bias	Facility	Error rate	Input	Thread (GBP)	Time (GBP)	RAM (GBP)	Recovered MSBs
NIST P-192	1-bit	AWS EC2	0	2^{29}	2304	113h	492GB	39
NIST P-192	1-bit	AWS EC2	1%	2^{35}	2304	52h	492GB	39
`sect163r1`	1-bit	Cluster	0	2^{23}	256	7h	80GB	36
`sect163r1`	1-bit	Workstation	2.7%	2^{24}	48	42h	250GB	35
`sect163r1`	2-bit	Workstation	0	1024	16	2h	96GB	32

Table 1: Computational results for the first round of Bleichenbacher

- Attack on **P-192** is made possible by our highly optimized parallel implementation.
- Attack on `sect163r1` requires much less signatures

Takeaways

- Improved analysis of **Blechenbacher's attack** to recover ECDSA/Schnorr secret keys

- Application: **LadderLeak**
  - Tiny timing leakage from the Montgomery ladder scalar multiplication in OpenSSL `1.0.2u` and `1.1.0l`
  - Coordinated disclosure: fixed in April 2020

- Interesting connection between the HNP and GBP

Subsequent Works & Future Directions

- [AH21] Feasibility of lattice attack against 1-bit leakage
- Further improvements to the data complexity?
- Other sources of small leakage?

Takeaways

- Improved analysis of **Blechenbacher's attack** to recover ECDSA/Schnorr secret keys
- Application: **LadderLeak**
  - Tiny timing leakage from the Montgomery ladder scalar multiplication in OpenSSL `1.0.2u` and `1.1.0l`
  - Coordinated disclosure: fixed in April 2020
- Interesting connection between the HNP and GBP

Subsequent Works & Future Directions

- **[AH21]** Feasibility of lattice attack against 1-bit leakage
- Further improvements to the data complexity?
- Other sources of small leakage?

Takeaways

- Improved analysis of Blechenbacher’s attack to recover ECDSA/Schnorr secret keys
- Application: **LadderLeak**
  - Tiny timing leakage from the Montgomery ladder scalar multiplication in OpenSSL `1.0.2u` and `1.1.0l`
  - Coordinated disclosure: fixed in April 2020
- Interesting connection between the HNP and GBP

Subsequent Works & Future Directions

- **[AH21]** Feasibility of lattice attack against 1-bit leakage
- Further improvements to the data complexity?
- Other sources of small leakage?

Takeaways

- Improved analysis of **Blechenbacher's attack** to recover ECDSA/Schnorr secret keys
- Application: **LadderLeak**
  - Tiny timing leakage from the Montgomery ladder scalar multiplication in OpenSSL `1.0.2u` and `1.1.0l`
  - Coordinated disclosure: fixed in April 2020
- Interesting connection between the HNP and GBP

Subsequent Works & Future Directions

- **[AH21]** Feasibility of lattice attack against 1-bit leakage
- Further improvements to the data complexity?
- Other sources of small leakage?

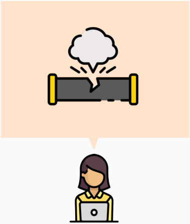
Takeaways

- Improved analysis of **Blechenbacher's attack** to recover ECDSA/Schnorr secret keys
- Application: **LadderLeak**
  - Tiny timing leakage from the Montgomery ladder scalar multiplication in OpenSSL `1.0.2u` and `1.1.0l`
  - Coordinated disclosure: fixed in April 2020
- Interesting connection between the HNP and GBP

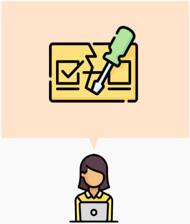
Subsequent Works & Future Directions

- [AH21] Feasibility of lattice attack against 1-bit leakage
- Further improvements to the data complexity?
- Other sources of small leakage?

Advanced Security

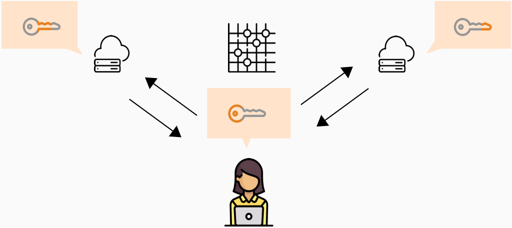


☑ What happens if the signer partially leaks randomness?



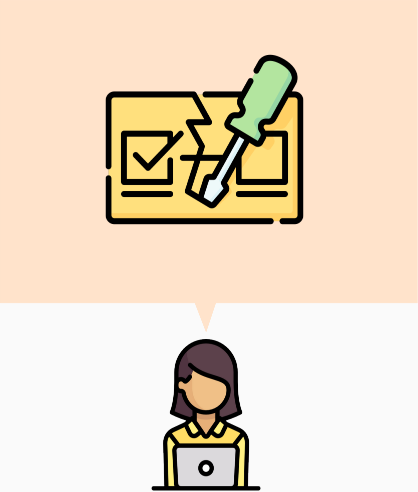
What happens if the signer produces faulty signatures?

New Constructions



Can we construct multi-party signatures from lattice ZK proof?

Can we add verifiability to ciphertexts using ZK proof?



What happens if the signer produces faulty signatures?

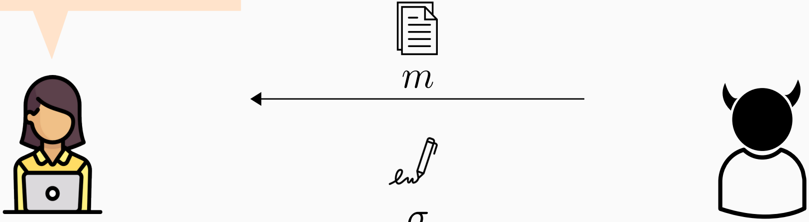
Popular Solution: Deterministic Randomness Generation

1. Randomized signature : $\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$ ☹ Risk of randomness bias!
2. Deterministic signature : $r := \mathsf{H}(sk, m)$

- Hash each message keyed with sk.
- Widely implemented, e.g. in EdDSA, ECDSA, Dilithium, etc.
- However, another practical issue arises...

Fault Attack Vulnerability of Deterministic Randomness

```
$r := \mathsf{H}(\mathsf{sk}, m)$
$(a, \mathsf{st}) := \mathsf{Com}(\mathsf{sk}, r)$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := \mathsf{Resp}(\mathsf{sk}, c, \mathsf{st})$
$\sigma := (a, z)$
```



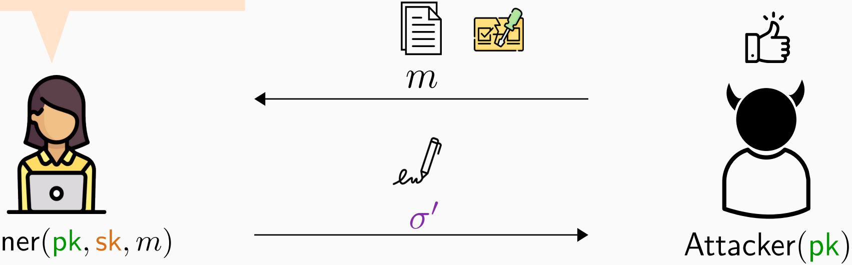
$\mathsf{Signer}(\mathsf{pk}, \mathsf{sk}, m)$

$\mathsf{Attacker}(\mathsf{pk})$

Fault Attack Vulnerability of Deterministic Randomness

```
$r := \mathsf{H}(\mathsf{sk}, m)$
$(a, \mathsf{st}) := \mathsf{Com}(\mathsf{sk}, r)$
$c' := \mathsf{H}(\mathsf{pk}, a, m)$
$z' := \mathsf{Resp}(\mathsf{sk}, c', \mathsf{st})$
$\sigma' := (a, z')$
```

- Tamper with the device to provoke randomness reuse
- Given (a, c, z) and (a, c', z'), sk can be recovered!
- cf. Special soundness



m

σ'

Signer(pk, sk, m)

Attacker(pk)

Better Countermeasure? – Randomness Hedging

1. Randomized signature : ~~$r \leftarrow \mathsf{RNG}(\cdot)$~~ ☹ Risk of randomness bias!
2. Deterministic signature : ~~$r := \mathsf{H}(sk, m)$~~ ☹ Vulnerable to fault attacks!
3. Hedged signature : $r := \mathsf{H}(sk, m, \mathsf{nonce})$ ☺ Seems secure?

- nonce: Number only used once
- nonce can be derived from low-quality RNG or counter
- r doesn't repeat on the same m.
- Seems secure, but no formal analysis so far.

Q. To what extent are hedged FS signatures secure against fault attacks?

Better Countermeasure? – Randomness Hedging

1. Randomized signature : $\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$ ☹ Risk of randomness bias!
2. Deterministic signature : $\cancel{r := \mathsf{H}(sk, m)}$ ☹ Vulnerable to fault attacks!
3. Hedged signature : $r := \mathsf{H}(sk, m, \mathsf{nonce})$ ☺ Seems secure?

- nonce: Number only used once
- nonce can be derived from low-quality RNG or counter
- r doesn't repeat on the same m.
- Seems secure, but no formal analysis so far.

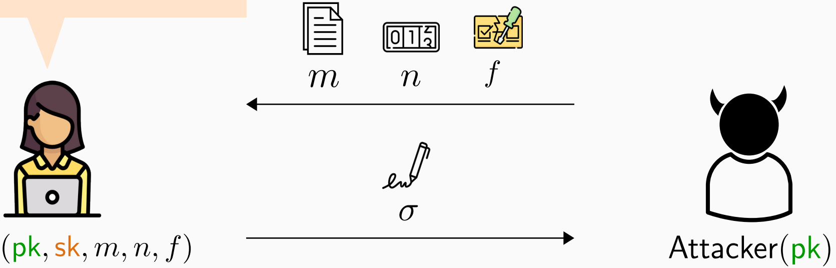
Q. To what extent are hedged FS signatures secure against fault attacks?

Our Fault Attacker Model

```
$r := \mathsf{H}(\mathsf{sk}, m, n)$
$(a, \mathsf{st}) := \mathsf{Com}(\mathsf{sk}, r)$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z := \mathsf{Resp}(\mathsf{sk}, c, \mathsf{st})$
$\sigma := (a, z)$
```

UF-FCMNA security

- UnForgeability against Faults, Chosen Message and Nonce Attacks
- Attacker can choose non-repeating n
- Attacker can inject f to intermediate computation
- $f \in \{\texttt{flip\_bit}, \texttt{set\_bit}\}$: 1-bit tampering function



Signer(pk, sk, m, n, f)

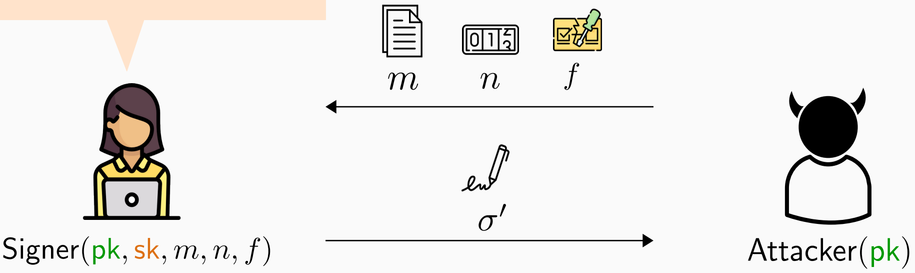
Attacker(pk)

Our Fault Attacker Model

```
$r := \mathsf{H}(\mathsf{sk}, m, n)$
$(a, \mathsf{st}) := \mathsf{Com}(\mathsf{sk}, r)$
$c' := f(\mathsf{H}(\mathsf{pk}, a, m))$
$z' := \mathsf{Resp}(\mathsf{sk}, c', \mathsf{st})$
$\sigma' := (a, z')$
```

UF-FCMNA security

- UnForgeability against Faults, Chosen Message and Nonce Attacks
- Attacker can choose non-repeating n
- Attacker can inject f to intermediate computation
- $f \in \{\texttt{flip\_bit}, \texttt{set\_bit}\}$: 1-bit tampering function

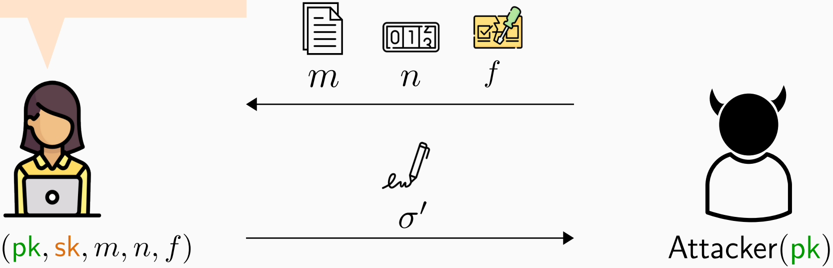


Our Fault Attacker Model

```
$r := \mathsf{H}(\mathsf{sk}, m, n)$
$(a, \mathsf{st}) := \mathsf{Com}(\mathsf{sk}, r)$
$c := \mathsf{H}(\mathsf{pk}, a, m)$
$z' := \mathsf{Resp}(\mathsf{sk}, c, f(\mathsf{st}))$
$\sigma' := (a, z')$
```

UF-FCMNA security

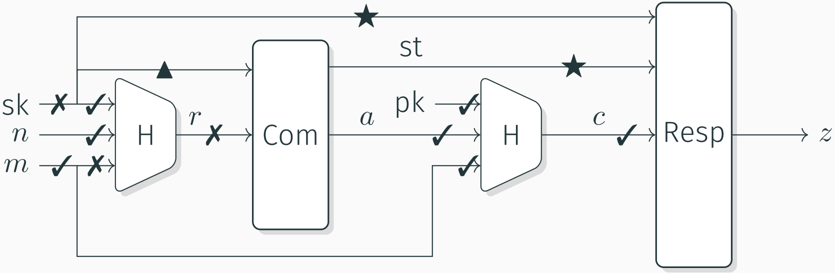
- UnForgeability against Faults, Chosen Message and Nonce Attacks
- Attacker can choose non-repeating n
- Attacker can inject f to intermediate computation
- $f \in \{\texttt{flip\_bit}, \texttt{set\_bit}\}$: 1-bit tampering function



$\mathsf{Signer}(\mathsf{pk}, \mathsf{sk}, m, n, f)$

$\mathsf{Attacker}(\mathsf{pk})$

Our Results (in the Random Oracle Model)



✓ Secure against single-bit flip/stuck-at faults.

✗ Insecure against single-bit flip/stuck-at faults.

★ Security only holds for signatures from subset-revealing ID (e.g. Picnic).

▲ Security only holds for signatures from input-delayed ID (e.g. XEdDSA).

Takeaways

- Formal attacker model and security notions to capture the corrupted nonces and bit-tampering faults
- Hedged FS signatures are provably more resilient than the randomized / deterministic FS
- Application
  - XEdDSA: Hedged variant of EdDSA used in Signal
  - Picnic: NIST PQC competition candidate

Concurrent/Subsequent Works & Future Directions

- **[FG20]** Multi-bit/position bit-flip faults
- **[GHHM21]** Lifted our result to the QROM
- Lattice signatures from FS with aborts?

Takeaways

- Formal attacker model and security notions to capture the corrupted nonces and bit-tampering faults
- Hedged FS signatures are provably more resilient than the randomized / deterministic FS
- Application
  - XEdDSA: Hedged variant of EdDSA used in Signal
  - Picnic: NIST PQC competition candidate

Concurrent/Subsequent Works & Future Directions

- **[FG20]** Multi-bit/position bit-flip faults
- **[GHHM21]** Lifted our result to the QROM
- Lattice signatures from FS with aborts?

Takeaways

- Formal attacker model and security notions to capture the corrupted nonces and bit-tampering faults
- Hedged FS signatures are provably more resilient than the randomized / deterministic FS
- Application
  - XEdDSA: Hedged variant of EdDSA used in Signal
  - Picnic: NIST PQC competition candidate

Concurrent/Subsequent Works & Future Directions

- [FG20] Multi-bit/position bit-flip faults
- [GHHM21] Lifted our result to the QROM
- Lattice signatures from FS with aborts?

Takeaways

- Formal attacker model and security notions to capture the corrupted nonces and bit-tampering faults
- Hedged FS signatures are provably more resilient than the randomized / deterministic FS
- Application
  - XEdDSA: Hedged variant of EdDSA used in Signal
  - Picnic: NIST PQC competition candidate

Concurrent/Subsequent Works & Future Directions

- **[FG20]** Multi-bit/position bit-flip faults
- **[GHHM21]** Lifted our result to the QROM
- Lattice signatures from FS with aborts?

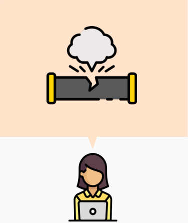
Takeaways

- Formal attacker model and security notions to capture the corrupted nonces and bit-tampering faults
- Hedged FS signatures are provably more resilient than the randomized / deterministic FS
- Application
    - XEdDSA: Hedged variant of EdDSA used in Signal
    - Picnic: NIST PQC competition candidate

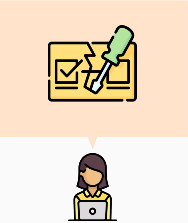
Concurrent/Subsequent Works & Future Directions

- **[FG20]** Multi-bit/position bit-flip faults
- **[GHHM21]** Lifted our result to the QROM
- Lattice signatures from FS with aborts?

Advanced Security

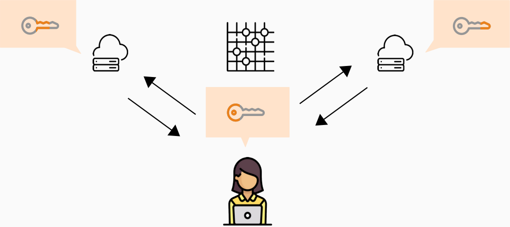


☑ What happens if the signer partially leaks randomness?



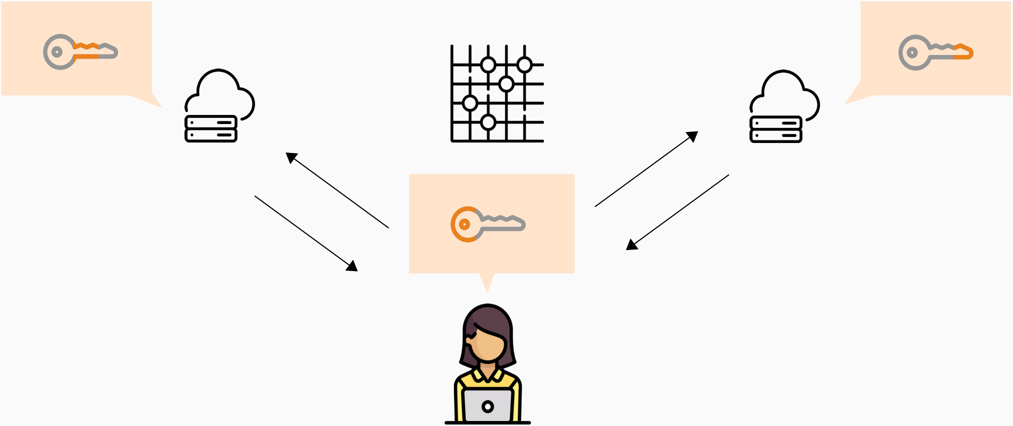
☑ What happens if the signer produces faulty signatures?

New Constructions



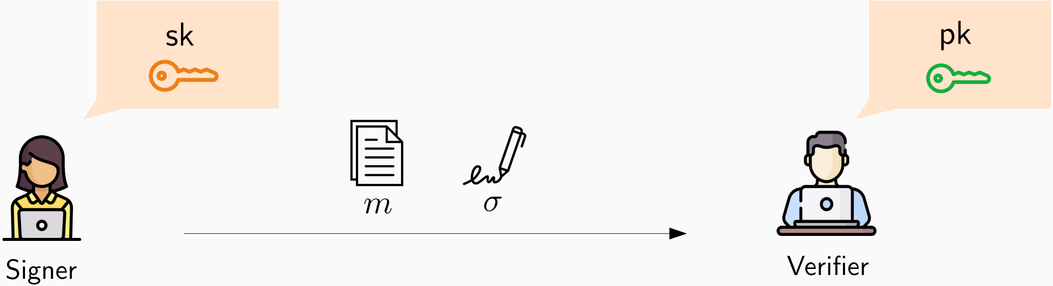
Can we construct multi-party signatures from lattice ZK proof?

Can we add verifiability to ciphertexts using ZK proof?



Can we construct multi-party signatures from lattice ZK proof?

Single-User Signature



sk

pk

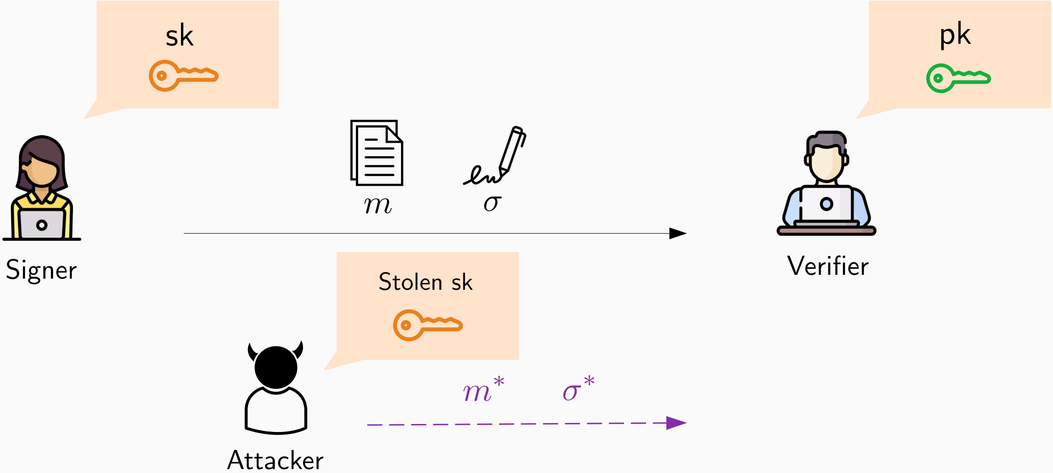
m

σ

Signer

Verifier

Single-User Signature



sk

Signer

m σ

pk

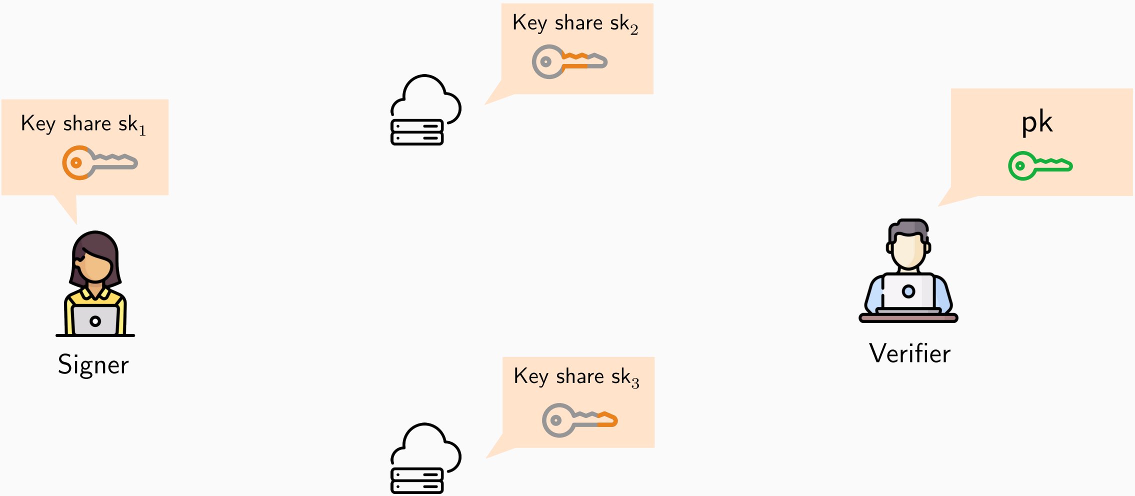
Verifier

Stolen sk

Attacker

m^* σ^*

Multi-Party Signature



Key share sk_1

Signer

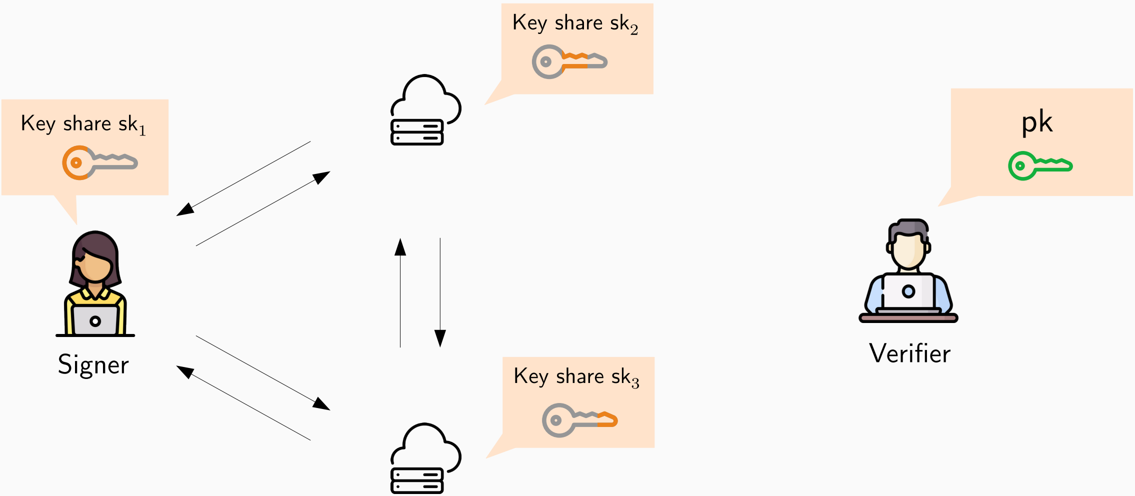
Key share sk_2

Key share sk_3

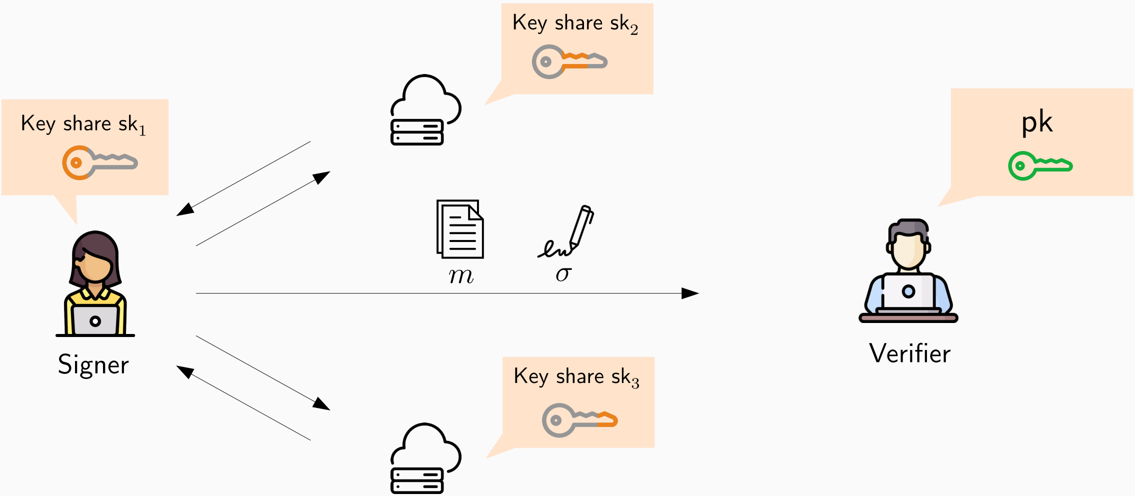
pk

Verifier

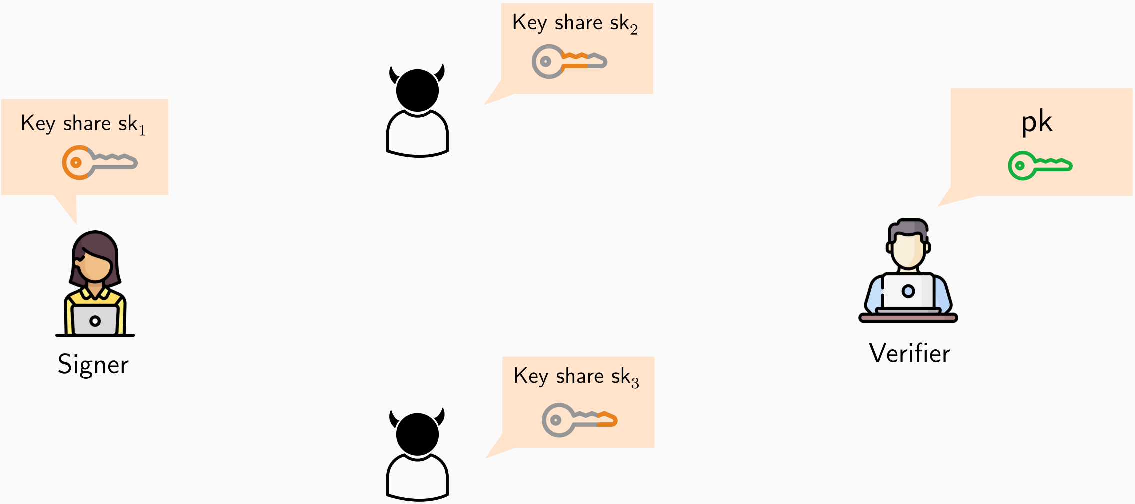
Multi-Party Signature


Key share sk_1
Signer
Key share sk_2
Key share sk_3
pk
Verifier

Multi-Party Signature


Key share sk_2
Key share sk_1
pk
m
σ
Signer
Verifier
Key share sk_3

Multi-Party Signature



Key share sk_2

Key share sk_1

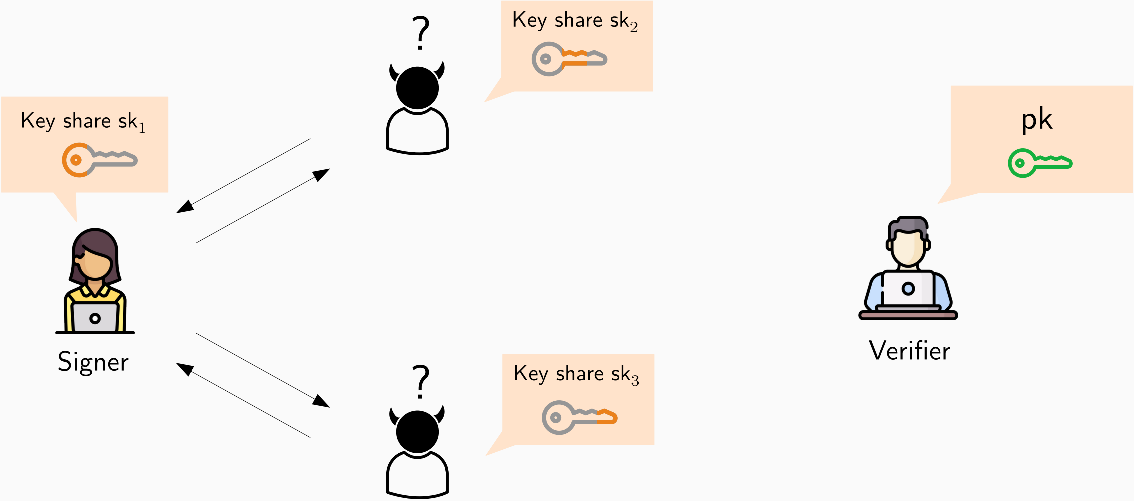
pk

Signer

Verifier

Key share sk_3

Multi-Party Signature



Key share sk_1

Signer

?

Key share sk_2

?

Key share sk_3

pk

Verifier

Landscape of Multi-Party Fiat-Shamir Signing

# Round	Method	Schnorr	Lattice
3	Commit&Open	BN06, MuSig,GJKR07,KMOS21,GKMN21,Lin22	ES16,FH20,BK20,DOTT-DS3
2	TD-Hom-Com	mBCJ, HBMS, MuSig-DN	DOTT-MS, DOTT-DS2
1 (Off) + 1 (On)	Linear Combination	MuSig2, DWMS, FROST	MuSig-L

- Orange: Multi-signature
- Green: Threshold signature (ours are only (n, n)-threshold)
- Fiat-Shamir with aborts (Lyubashevsky '09/'12) $\approx$ Lattice-based Schnorr

*Q. Can we construct **two-round**, provably secure schemes from lattices?*

Landscape of Multi-Party Fiat-Shamir Signing

# Round	Method	Schnorr	Lattice
3	Commit&Open	BN06, MuSig,GJKR07,KMOS21,GKMN21,Lin22	ES16,FH20,BK20,DOTT-DS3
2	TD-Hom-Com	mBCJ, HBMS, MuSig-DN	DOTT-MS, DOTT-DS2
1 (Off) + 1 (On)	Linear Combination	MuSig2, DWMS, FROST	MuSig-L

- Orange: Multi-signature
- Green: Threshold signature (ours are only (n, n)-threshold)
- Fiat-Shamir with aborts (Lyubashevsky '09/'12) $\approx$ Lattice-based Schnorr

Q. Can we construct ***two-round****, provably secure schemes from lattices?*

Landscape of Multi-Party Fiat-Shamir Signing

# Round	Method	Schnorr	Lattice
3	Commit&Open	BN06, MuSig,GJKR07,KMOS21,GKMN21,Lin22	ES16,FH20,BK20,DOTT-DS3
2	TD-Hom-Com	mBCJ, HBMS, MuSig-DN	DOTT-MS, DOTT-DS2
1 (Off) + 1 (On)	Linear Combination	MuSig2, DWMS, FROST	MuSig-L

- Orange: Multi-signature
- Green: Threshold signature (ours are only (n, n)-threshold)
- Fiat-Shamir with aborts (Lyubashevsky '09/'12) ≈ Lattice-based Schnorr

*Q. Can we construct **two-round**, provably secure schemes from lattices?*

Schnorr vs Fiat-Shamir with Aborts

Schnorr ID

$r \leftarrow_{\$} \mathbb{Z}_q$

$a := g^r \quad \xrightarrow{\quad a \quad}$

$\xleftarrow{\quad c \quad} \quad c \leftarrow_{\$} \mathbb{Z}_q$

Accept iff

$z := c \cdot \mathsf{sk} + r \quad \xrightarrow{\quad z \quad} \quad g^z = \mathsf{pk}^c \cdot a$

Prover($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}$)

Verifier(pk)

Schnorr vs Fiat-Shamir with Aborts

Schnorr ID

$r \leftarrow_{\$} \mathbb{Z}_q$

$a := g^r$ $\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad c \quad}$ $c \leftarrow_{\$} \mathbb{Z}_q$

Accept iff

$z := c \cdot \mathsf{sk} + r$ $\xrightarrow{\quad z \quad}$ $g^z = \mathsf{pk}^c \cdot a$

Prover($\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk}$) Verifier($\mathsf{pk}$)

Fiat-Shamir with Aborts ID

$\mathbf{r} \leftarrow_{\$} D$

$\mathbf{u} := \mathbf{A}\mathbf{r}$ $\xrightarrow{\quad \mathbf{u} \quad}$

$\xleftarrow{\quad c \quad}$ $c \leftarrow_{\$} C \subset R$

$\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$

If RejSamp($\mathbf{z}$) $= 0$:

Abort

$\xrightarrow{\quad \mathbf{z} \quad}$

Accept iff

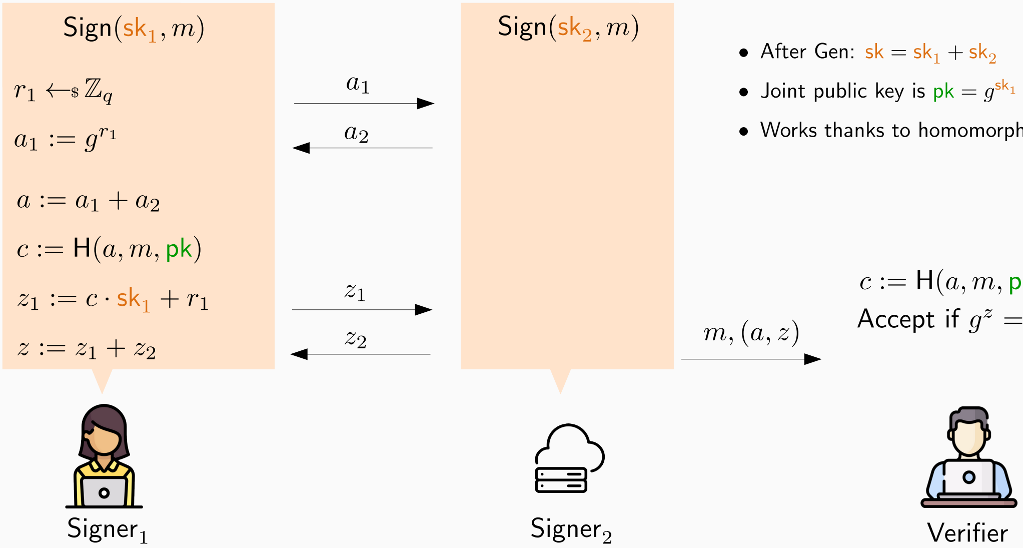
$\mathbf{A}\mathbf{z} = c \cdot \mathsf{pk} + \mathbf{u}$

$\wedge \ \|\mathbf{z}\| \leq B$

Prover($\mathsf{pk} = \mathbf{A} \cdot \mathsf{sk}, \mathsf{sk}$) Verifier($\mathsf{pk}$)

Bare-Bones Two-Party Schnorr

- After Gen: $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$
- Joint public key is $\mathsf{pk} = g^{\mathsf{sk}_1} \cdot g^{\mathsf{sk}_2}$
- Works thanks to homomorphism of g^x



$\mathsf{Sign}(\mathsf{sk}_1, m)$

$r_1 \leftarrow_\$ \mathbb{Z}_q$

$a_1 := g^{r_1}$

$a := a_1 + a_2$

$c := \mathsf{H}(a, m, \mathsf{pk})$

$z_1 := c \cdot \mathsf{sk}_1 + r_1$

$z := z_1 + z_2$

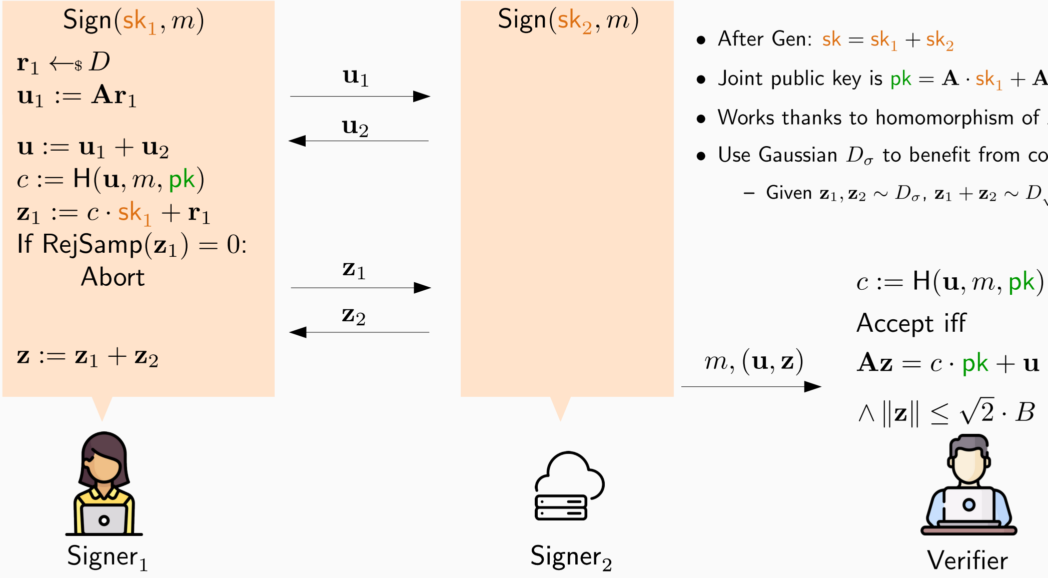
$\mathsf{Sign}(\mathsf{sk}_2, m)$

$c := \mathsf{H}(a, m, \mathsf{pk})$

Accept if $g^z = \mathsf{pk}^c \cdot a$

Bare-Bones Two-Party Signing from Lattices

- After Gen: $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$
- Joint public key is $\mathsf{pk} = \mathbf{A} \cdot \mathsf{sk}_1 + \mathbf{A} \cdot \mathsf{sk}_2$
- Works thanks to homomorphism of $\mathbf{A} \cdot \mathbf{x}$
- Use Gaussian D_σ to benefit from convolution:
  - Given $\mathbf{z}_1, \mathbf{z}_2 \sim D_\sigma$, $\mathbf{z}_1 + \mathbf{z}_2 \sim D_{\sqrt{2} \cdot \sigma}$



$\mathsf{Sign}(\mathsf{sk}_1, m)$

```
$\mathbf{r}_1 \leftarrow_\$ D$
$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\mathbf{u} := \mathbf{u}_1 + \mathbf{u}_2$
$c := \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$
$\mathbf{z}_1 := c \cdot \mathsf{sk}_1 + \mathbf{r}_1$
If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:
    Abort

$\mathbf{z} := \mathbf{z}_1 + \mathbf{z}_2$
```

$\mathsf{Sign}(\mathsf{sk}_2, m)$

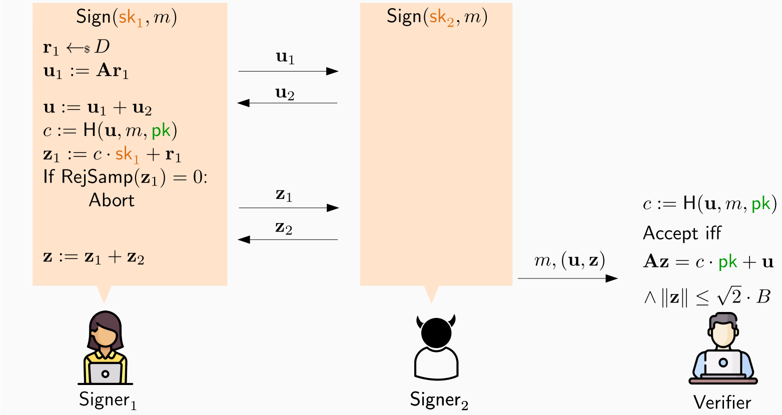
$c := \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$

Accept iff

$\mathbf{A}\mathbf{z} = c \cdot \mathsf{pk} + \mathbf{u}$

$\wedge \|\mathbf{z}\| \leq \sqrt{2} \cdot B$

Issues of Bare-Bones Protocols



Sign(sk_1, m)

```
$\mathbf{r}_1 \leftarrow_\$ D$
$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\mathbf{u} := \mathbf{u}_1 + \mathbf{u}_2$
$c := \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$
$\mathbf{z}_1 := c \cdot \mathsf{sk}_1 + \mathbf{r}_1$
If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:
        Abort

$\mathbf{z} := \mathbf{z}_1 + \mathbf{z}_2$
```

$\xrightarrow{\mathbf{u}_1}$

$\xleftarrow{\mathbf{u}_2}$

$\xrightarrow{\mathbf{z}_1}$

$\xleftarrow{\mathbf{z}_2}$

Sign(sk_2, m)

$\xrightarrow{m, (\mathbf{u}, \mathbf{z})}$

$c := \mathsf{H}(\mathbf{u}, m, \mathsf{pk})$

Accept iff

$\mathbf{A}\mathbf{z} = c \cdot \mathsf{pk} + \mathbf{u}$

$\wedge \|\mathbf{z}\| \leq \sqrt{2} \cdot B$

Signer$_1$

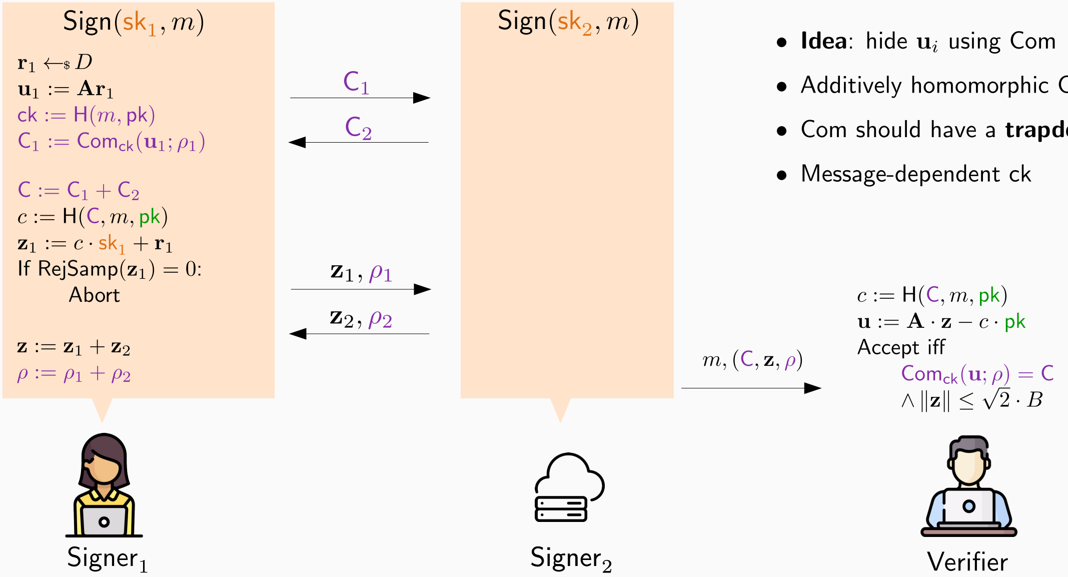
Signer$_2$

Verifier

1. Malicious Signer$_2$ can choose $\mathbf{u}_2$ depending $\mathbf{u}_1$
    - Forgery attack in the **concurrent setting** (Drijvers et al.'19)
2. Simulation of rejected $(\mathbf{u}_i, c, \perp)$
    - Underlying ID scheme is only HVZK for non-aborting transcripts

Our Solution

- **Idea**: hide $\mathbf{u}_i$ using Com
- Additively homomorphic Com
- Com should have a **trapdoor**
- Message-dependent ck



$\mathsf{Sign}(\mathsf{sk}_1, m)$

$\mathbf{r}_1 \leftarrow_\$ D$

$\mathbf{u}_1 := \mathbf{A}\mathbf{r}_1$

$\mathsf{ck} := \mathsf{H}(m, \mathsf{pk})$

$\mathsf{C}_1 := \mathsf{Com}_{\mathsf{ck}}(\mathbf{u}_1; \rho_1)$

$\mathsf{C} := \mathsf{C}_1 + \mathsf{C}_2$

$c := \mathsf{H}(\mathsf{C}, m, \mathsf{pk})$

$\mathbf{z}_1 := c \cdot \mathsf{sk}_1 + \mathbf{r}_1$

If $\mathsf{RejSamp}(\mathbf{z}_1) = 0$:

Abort

$\mathbf{z} := \mathbf{z}_1 + \mathbf{z}_2$

$\rho := \rho_1 + \rho_2$

C_1

C_2

$\mathbf{z}_1, \rho_1$

$\mathbf{z}_2, \rho_2$

$\mathsf{Sign}(\mathsf{sk}_2, m)$

$m, (\mathsf{C}, \mathbf{z}, \rho)$

$c := \mathsf{H}(\mathsf{C}, m, \mathsf{pk})$

$\mathbf{u} := \mathbf{A} \cdot \mathbf{z} - c \cdot \mathsf{pk}$

Accept iff

$\mathsf{Com}_{\mathsf{ck}}(\mathbf{u}; \rho) = \mathsf{C}$

$\wedge \|\mathbf{z}\| \leq \sqrt{2} \cdot B$

Takeaways

- **Two-round** multi-party signing from lattices
  - **n-out-of-n threshold signature**
  - **Multi-signature**
- Proof in the (classical) ROM from the standard SIS and LWE assumptions
- Subtlety of lifting DLog schemes to the lattice world

Subsequent Works & Future Directions

- **MuSig-L [BTT22]** Single-round online phase
- Efficient implementation
- Proof in the QROM
- t-of-n signature from lattices

Takeaways

- **Two-round** multi-party signing from lattices
  - **n-out-of-n threshold signature**
  - **Multi-signature**
- Proof in the (classical) ROM from the standard SIS and LWE assumptions
- Subtlety of lifting DLog schemes to the lattice world

Subsequent Works & Future Directions

- **MuSig-L [BTT22]** Single-round online phase
- Efficient implementation
- Proof in the QROM
- t-of-n signature from lattices

Takeaways

- **Two-round** multi-party signing from lattices
  - **n-out-of-n threshold signature**
  - **Multi-signature**
- Proof in the (classical) ROM from the standard SIS and LWE assumptions
- Subtlety of lifting DLog schemes to the lattice world

Subsequent Works & Future Directions

- **MuSig-L [BTT22]** Single-round online phase
- Efficient implementation
- Proof in the QROM
- t-of-n signature from lattices

Takeaways

- **Two-round** multi-party signing from lattices
    - **n-out-of-n threshold signature**
    - **Multi-signature**
- Proof in the (classical) ROM from the standard SIS and LWE assumptions
- Subtlety of lifting DLog schemes to the lattice world

Subsequent Works & Future Directions

- **MuSig-L [BTT22]** Single-round online phase
- Efficient implementation
- Proof in the QROM
- t-of-n signature from lattices

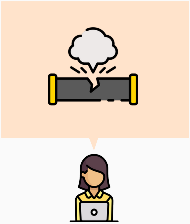
Takeaways

- **Two-round** multi-party signing from lattices
    - **n-out-of-n threshold signature**
    - **Multi-signature**
- Proof in the (classical) ROM from the standard SIS and LWE assumptions
- Subtlety of lifting DLog schemes to the lattice world

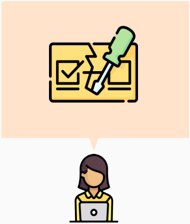
Subsequent Works & Future Directions

- **MuSig-L [BTT22]** Single-round online phase
- Efficient implementation
- Proof in the QROM
- t-of-n signature from lattices

Advanced Security

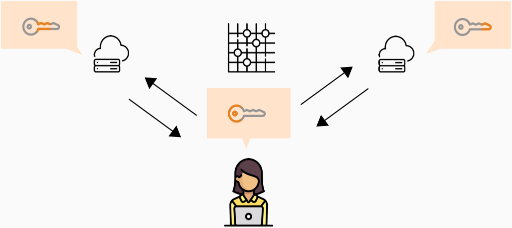


✓ What happens if the signer partially leaks randomness?



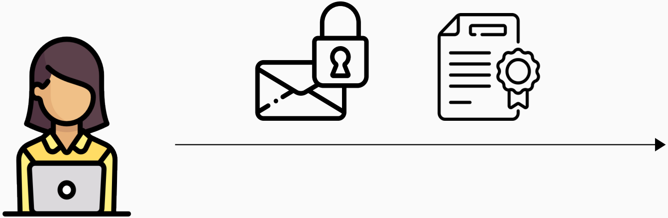
✓ What happens if the signer produces faulty signatures?

New Constructions



✓ Can we construct multi-party signatures from lattice ZK proof?

Can we add verifiability to ciphertexts using ZK proof?



Can we add verifiability to ciphertexts using ZK proof?

Auditing Ciphertext Integrity

$\mathsf{Sender}(\mathsf{pk}, w)$

$\mathsf{Receiver}(\mathsf{sk})$

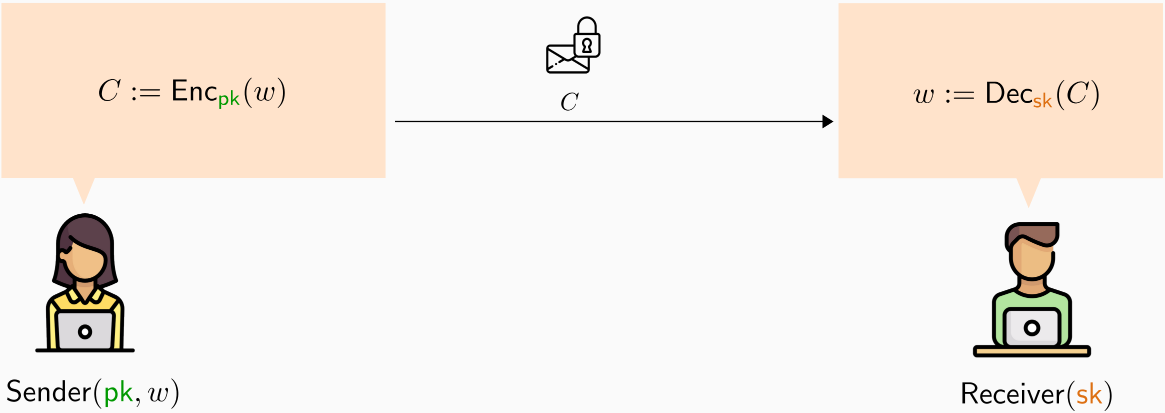
Auditing Ciphertext Integrity

$C := \mathsf{Enc}_{\mathsf{pk}}(w)$

$\mathsf{Sender}(\mathsf{pk}, w)$

$\mathsf{Receiver}(\mathsf{sk})$

Auditing Ciphertext Integrity



$C := \mathsf{Enc}_{\mathsf{pk}}(w)$

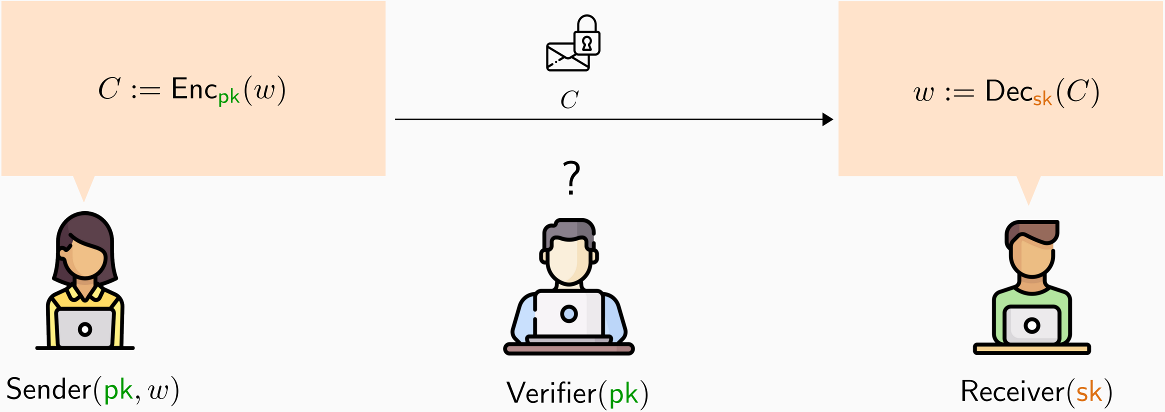
C

$w := \mathsf{Dec}_{\mathsf{sk}}(C)$

$\mathsf{Sender}(\mathsf{pk}, w)$

$\mathsf{Receiver}(\mathsf{sk})$

Auditing Ciphertext Integrity



$C := \mathsf{Enc}_{\mathsf{pk}}(w)$

C

?

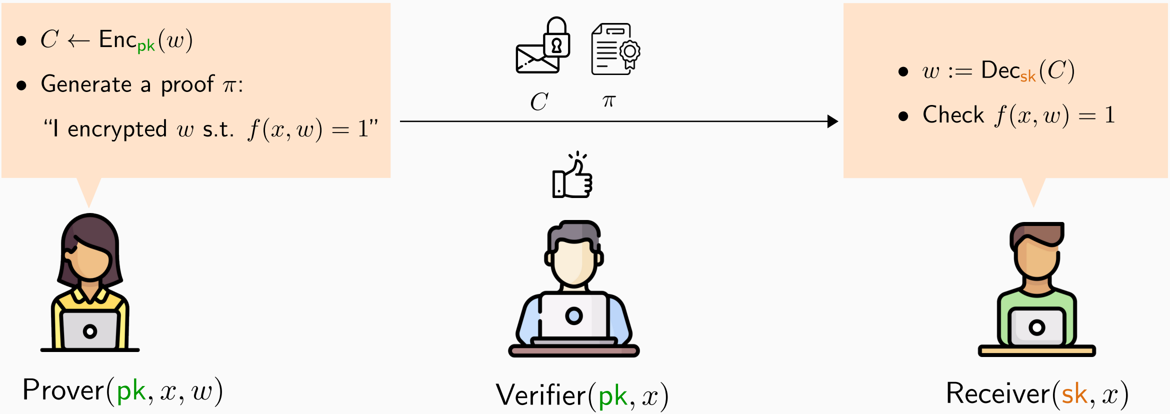
$w := \mathsf{Dec}_{\mathsf{sk}}(C)$

Sender(pk, w)

Verifier(pk)

Receiver(sk)

Verifiable Encryption



- $C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(w)$
- Generate a proof π:

  “I encrypted w s.t. $f(x, w) = 1$”

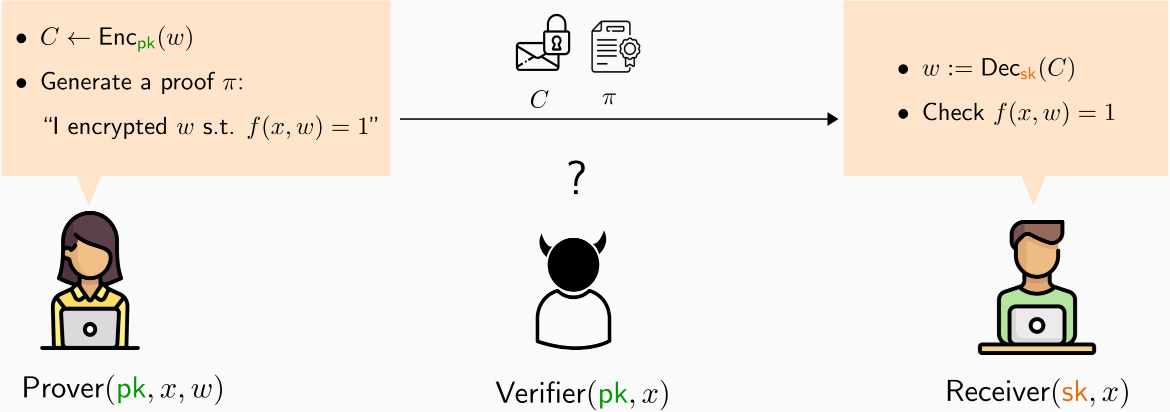
- $w := \mathsf{Dec}_{\mathsf{sk}}(C)$
- Check $f(x, w) = 1$

Prover(pk, x, w)

Verifier(pk, x)

Receiver(sk, x)

Zero Knowledge



- $C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(w)$
- Generate a proof π:

  "I encrypted w s.t. $f(x, w) = 1$"

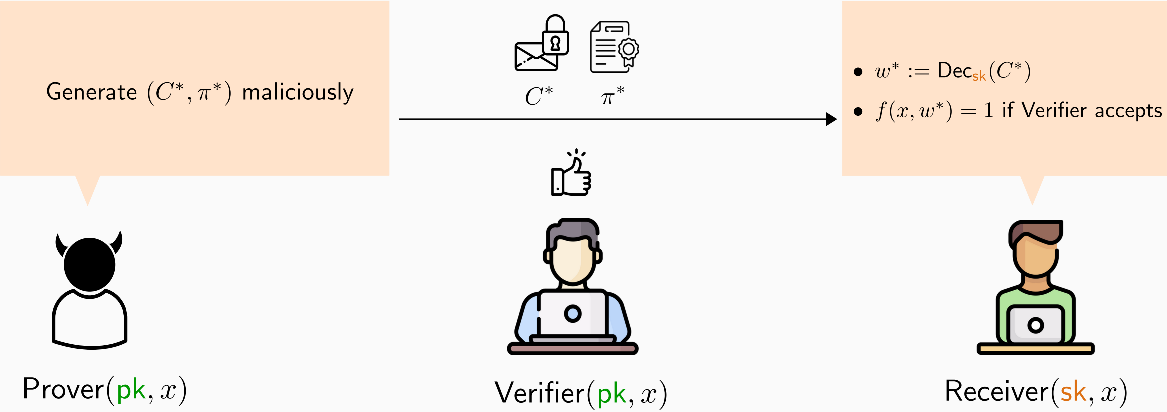
- $w := \mathsf{Dec}_{\mathsf{sk}}(C)$
- Check $f(x, w) = 1$

$\mathsf{Prover}(\mathsf{pk}, x, w)$

$\mathsf{Verifier}(\mathsf{pk}, x)$

$\mathsf{Receiver}(\mathsf{sk}, x)$

Validity



Generate (C^*, π^*) maliciously

C^* π^*

- $w^* := \mathsf{Dec}_{\mathsf{sk}}(C^*)$
- $f(x, w^*) = 1$ if Verifier accepts

Prover(pk, x)

Verifier(pk, x)

Receiver(sk, x)

Landscape of VE Constructions

	Generality of f	Ciphertext	Assumption
Camenisch–Shoup [CS03]	DL in $\mathbb{F}^*$ or $\mathbb{Z}_n^*$	Paillier	DCR
MuSig-DN [NRSW20]	DL	Elgamal	DDH
Lyubashevsky–Neven [LN17]	Linear relation	LPR	SIS/LWE
SAVER [LCKO19]	Any w/ SNARK	Elgamal	q-KEA
Beullens et al. [BDK$^+$21]	Membership in ring	Elgamal-like	DCSIDH
Camenisch–Damgård [CD00]	Any w/ Σ-protocol of 1-bit Ch.	PKE + Transcript	Undeniable IND-CPA PKE
Our result [TZ22]	Any w/ MPCitH ZKP	PKE + Transcript	Undeniable IND-CPA PKE

- Generality of relation f
- Flexibility in the receiver's PKE
- Minimizing assumptions

*Q. Can we construct **generic** VE supporting many f and PKE?*

Landscape of VE Constructions

	Generality of f	Ciphertext	Assumption
Camenisch–Shoup [CS03]	DL in $\mathbb{F}^*$ or $\mathbb{Z}_n^*$	Paillier	DCR
MuSig-DN [NRSW20]	DL	Elgamal	DDH
Lyubashevsky–Neven [LN17]	Linear relation	LPR	SIS/LWE
SAVER [LCKO19]	Any w/ SNARK	Elgamal	q-KEA
Beullens et al. [BDK$^+$21]	Membership in ring	Elgamal-like	DCSIDH
Camenisch–Damgård [CD00]	Any w/ Σ-protocol of 1-bit Ch.	PKE + Transcript	Undeniable IND-CPA PKE
Our result [TZ22]	Any w/ MPCitH ZKP	PKE + Transcript	Undeniable IND-CPA PKE

- Generality of relation f
- Flexibility in the receiver's PKE
- Minimizing assumptions

*Q. Can we construct **generic** VE supporting many f and PKE?*

Landscape of VE Constructions

	Generality of f	Ciphertext	Assumption
Camenisch–Shoup [CS03]	DL in $\mathbb{F}^*$ or $\mathbb{Z}_n^*$	Paillier	DCR
MuSig-DN [NRSW20]	DL	Elgamal	DDH
Lyubashevsky–Neven [LN17]	Linear relation	LPR	SIS/LWE
SAVER [LCKO19]	Any w/ SNARK	Elgamal	q-KEA
Beullens et al. [BDK+21]	Membership in ring	Elgamal-like	DCSIDH
Camenisch–Damgård [CD00]	Any w/ Σ-protocol of 1-bit Ch.	PKE + Transcript	Undeniable IND-CPA PKE
Our result [TZ22]	Any w/ MPCitH ZKP	PKE + Transcript	Undeniable IND-CPA PKE

- Generality of relation f
- Flexibility in the receiver’s PKE
- Minimizing assumptions

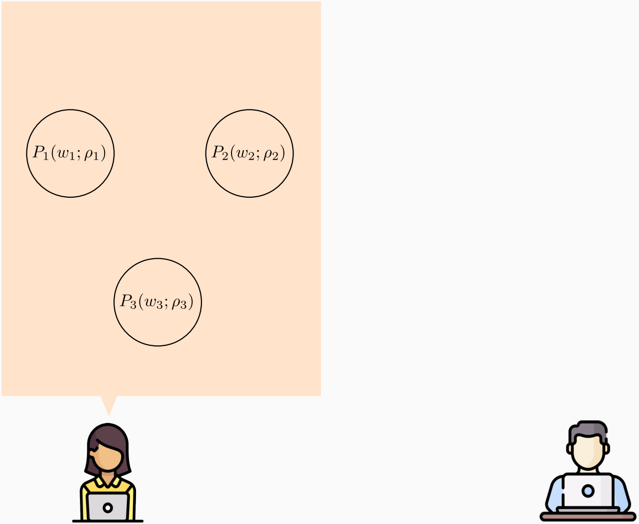
Q. Can we construct ***generic*** *VE supporting many f and PKE?*

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]

Prover(x, w)

Verifier(x)

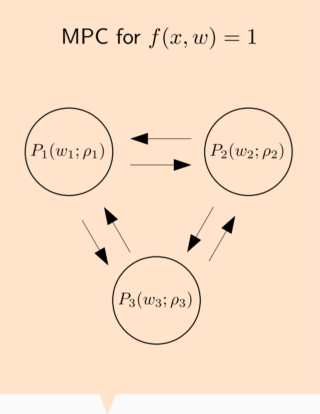
Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]



Prover(x, w)

Verifier(x)

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]

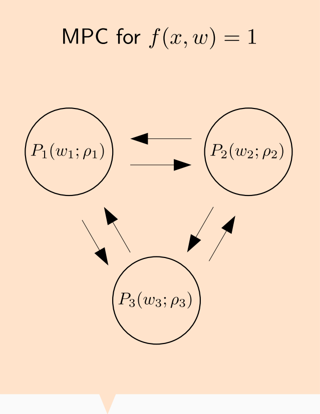


MPC for $f(x, w) = 1$

Prover(x, w)

Verifier(x)

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]



MPC for $f(x, w) = 1$

$P_1(w_1; \rho_1)$

$P_2(w_2; \rho_2)$

$P_3(w_3; \rho_3)$

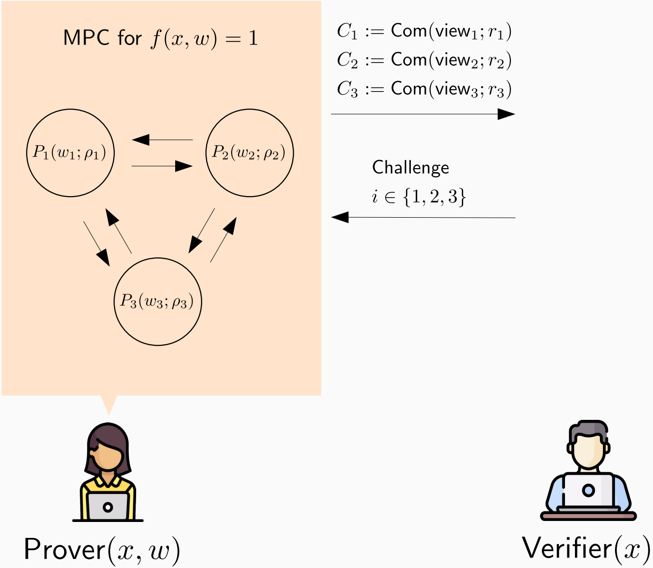
$$\begin{aligned} C_1 &:= \mathsf{Com}(\mathsf{view}_1; r_1) \\ C_2 &:= \mathsf{Com}(\mathsf{view}_2; r_2) \\ C_3 &:= \mathsf{Com}(\mathsf{view}_3; r_3) \end{aligned}$$

$$\xrightarrow{\hspace{4cm}}$$

Prover(x, w)

Verifier(x)

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]



MPC for $f(x, w) = 1$

$P_1(w_1; \rho_1)$

$P_2(w_2; \rho_2)$

$P_3(w_3; \rho_3)$

$C_1 := \mathsf{Com}(\mathsf{view}_1; r_1)$

$C_2 := \mathsf{Com}(\mathsf{view}_2; r_2)$

$C_3 := \mathsf{Com}(\mathsf{view}_3; r_3)$

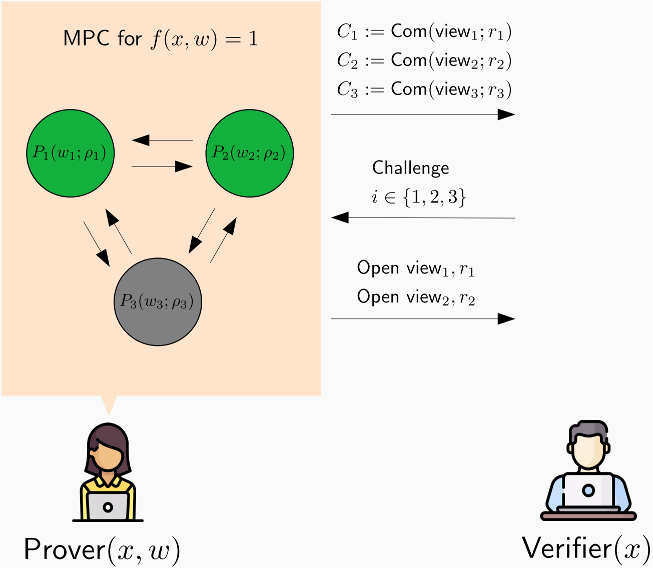
Challenge

$i \in \{1, 2, 3\}$

Prover(x, w)

Verifier(x)

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]



MPC for $f(x, w) = 1$

$P_1(w_1; \rho_1)$

$P_2(w_2; \rho_2)$

$P_3(w_3; \rho_3)$

$C_1 := \mathsf{Com}(\mathsf{view}_1; r_1)$

$C_2 := \mathsf{Com}(\mathsf{view}_2; r_2)$

$C_3 := \mathsf{Com}(\mathsf{view}_3; r_3)$

Challenge

$i \in \{1, 2, 3\}$

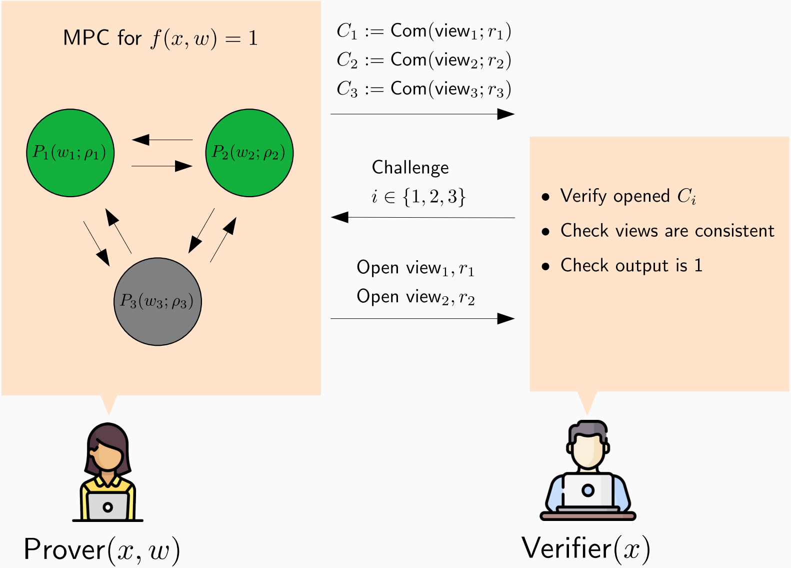
Open view_1, r_1

Open view_2, r_2

Prover(x, w)

Verifier(x)

Zero-knowledge Proof using MPC-in-the-head [IKOS07, GMO16]



MPC for $f(x, w) = 1$

$P_1(w_1; \rho_1)$

$P_2(w_2; \rho_2)$

$P_3(w_3; \rho_3)$

$C_1 := \mathsf{Com}(\mathsf{view}_1; r_1)$

$C_2 := \mathsf{Com}(\mathsf{view}_2; r_2)$

$C_3 := \mathsf{Com}(\mathsf{view}_3; r_3)$

Challenge

$i \in \{1, 2, 3\}$

Open view_1, r_1

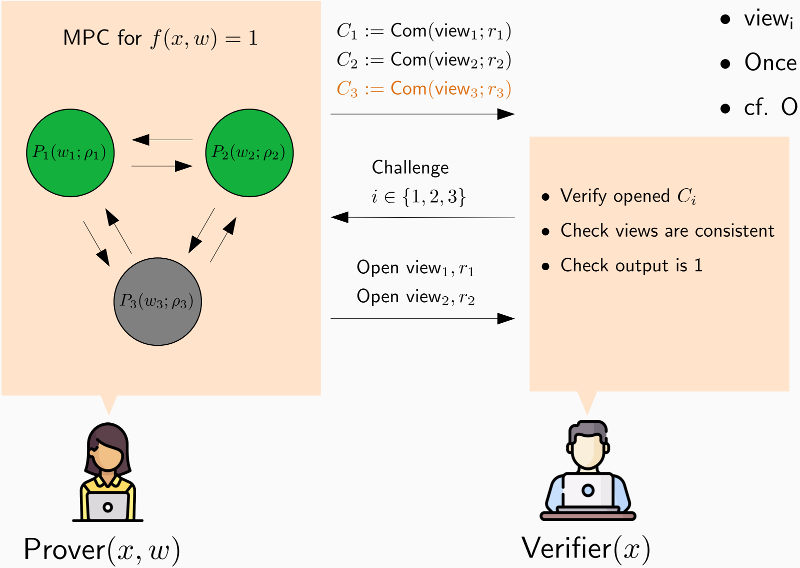
Open view_2, r_2

- Verify opened C_i
- Check views are consistent
- Check output is 1

$\mathsf{Prover}(x, w)$

$\mathsf{Verifier}(x)$

Observation



MPC for $f(x, w) = 1$

$P_1(w_1; \rho_1)$

$P_2(w_2; \rho_2)$

$P_3(w_3; \rho_3)$

$C_1 := \mathsf{Com}(\mathsf{view}_1; r_1)$

$C_2 := \mathsf{Com}(\mathsf{view}_2; r_2)$

$C_3 := \mathsf{Com}(\mathsf{view}_3; r_3)$

Challenge

$i \in \{1, 2, 3\}$

Open view_1, r_1

Open view_2, r_2

- Verify opened C_i
- Check views are consistent
- Check output is 1

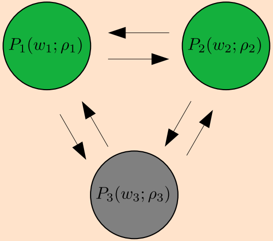
Prover(x, w)

Verifier(x)

- view_i contains a witness share w_i
- Once C_3 is opened, w can be recovered!
- cf. Online/straight-line extractability

Our Compiler for Verifiable Encryption: High-level Idea

MPC for $f(x, w) = 1$



$C_1 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_1; r_1)$
$C_2 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_2; r_2)$
$C_3 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_3; r_3)$ →

← Challenge $i \in \{1, 2, 3\}$

Open view_1, r_1
Open view_2, r_2 →

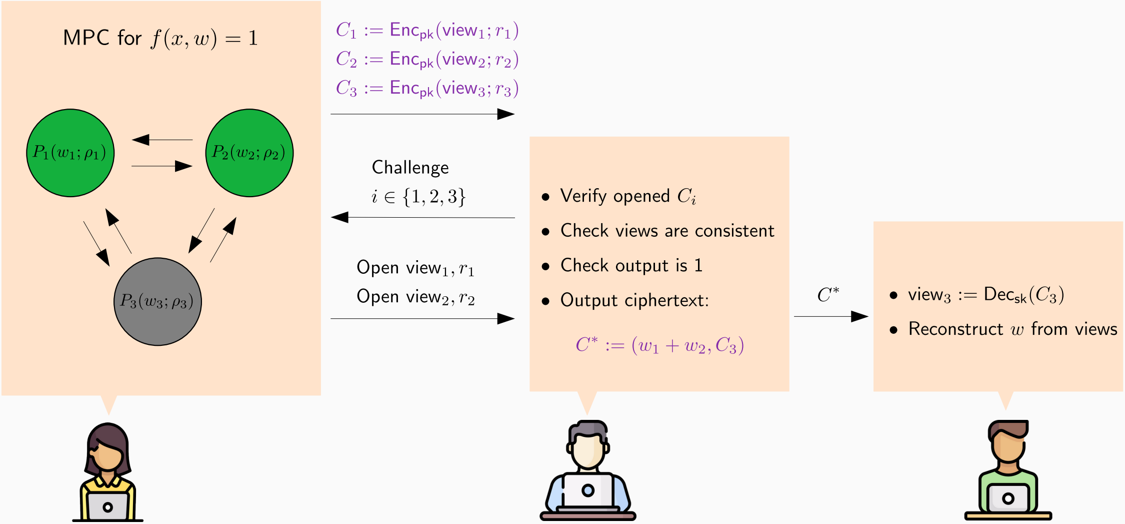
- Verify opened C_i
- Check views are consistent
- Check output is 1
- Output ciphertext:

$$C^* := (w_1 + w_2, C_3)$$

Prover(pk, x, w)

Verifier(pk, x)

Our Compiler for Verifiable Encryption: High-level Idea



MPC for $f(x, w) = 1$

$C_1 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_1; r_1)$
$C_2 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_2; r_2)$
$C_3 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_3; r_3)$

Challenge
$i \in \{1, 2, 3\}$

Open view_1, r_1
Open view_2, r_2

- Verify opened C_i
- Check views are consistent
- Check output is 1
- Output ciphertext:

$$C^* := (w_1 + w_2, C_3)$$

C^*

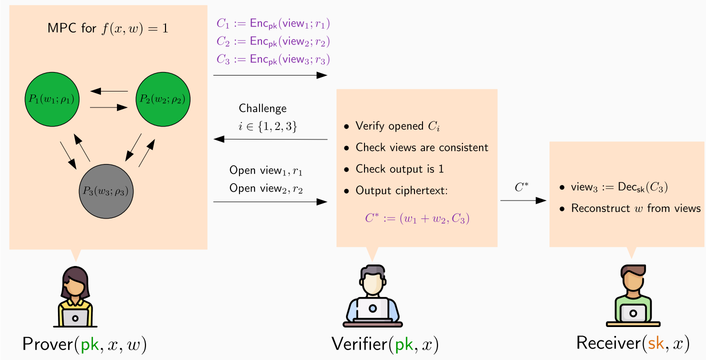
- $\mathsf{view}_3 := \mathsf{Dec}_{\mathsf{sk}}(C_3)$
- Reconstruct w from views

Prover(pk, x, w)

Verifier(pk, x)

Receiver(sk, x)

Security


MPC for $f(x,w)=1$
$P_1(w_1;\rho_1)$
$P_2(w_2;\rho_2)$
$P_3(w_3;\rho_3)$
$C_1 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_1; r_1)$
$C_2 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_2; r_2)$
$C_3 := \mathsf{Enc}_{\mathsf{pk}}(\mathsf{view}_3; r_3)$
Challenge
$i \in \{1,2,3\}$
Open view_1, r_1
Open view_2, r_2
• Verify opened C_i
• Check views are consistent
• Check output is 1
• Output ciphertext:
$C^* := (w_1 + w_2, C_3)$
C^*
• $\mathsf{view}_3 := \mathsf{Dec}_{\mathsf{sk}}(C_3)$
• Reconstruct w from views
Prover(pk, x, w)
Verifier(pk, x)
Receiver(sk, x)

- **Zero knowledge**: Follows from IND-CPA of $\mathsf{Enc}_{\mathsf{pk}}()$
- **Validity**: Follows from undeniability of $\mathsf{Enc}_{\mathsf{pk}}()$
  - Parallel repetitions to achieve negligible validity error

Interesting Corollaries

IKOS

- Verifiably encrypt witness for any NP relation

ZKBoo, KKW, Limbo

- Practical proofs for any circuit
- Encrypt Picnic private keys, hash function preimage, etc.

Banquet

- "I encrypted K such that $\mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})$" for public $(\mathsf{ct}, \mathsf{pt})$
- **Banquet** + **PQ-PKE** $\in \{$**Kyber**$,$ **FrodoKEM**$, \ldots\}$ = **Post-Quantum VE**

Distributed Key Generation in the Head (new)

- "I encrypted w such that $x = g^w$" for public x
- **Idea** Prover runs simple, passively secure DKG: $x := \prod_i g^{w_i}$

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
  - No proof-of-plaintext-knowledge
  - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
  1. DLog private keys
  2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable ZK** and **commit-and-prove ZK**?
- Compiling other IOPs into VE?

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
  - No proof-of-plaintext-knowledge
  - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
  1. DLog private keys
  2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable** ZK and **commit-and-prove** ZK?
- Compiling other IOPs into VE?

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
  - No proof-of-plaintext-knowledge
  - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
  1. DLog private keys
  2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable** ZK and **commit-and-prove** ZK?
- Compiling other IOPs into VE?

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
  - No proof-of-plaintext-knowledge
  - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
  1. DLog private keys
  2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable ZK** and **commit-and-prove ZK**?
- Compiling other IOPs into VE?

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
  - No proof-of-plaintext-knowledge
  - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
  1. DLog private keys
  2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable ZK** and **commit-and-prove ZK**?
- Compiling other IOPs into VE?

Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
    - No proof-of-plaintext-knowledge
    - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
    1. DLog private keys
    2. AES private keys

Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable ZK** and **commit-and-prove ZK**?
- Compiling other IOPs into VE?

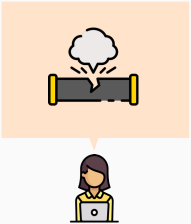
Takeaways

- Versatile VE for a large class of relations and PKE
- Performance is okay if efficient MPCitH exists for $f(x, w) \overset{?}{=} 1$
    - No proof-of-plaintext-knowledge
    - Possible improvements similar to improvements to MPCitH signatures
- Two concrete instantiations:
    1. DLog private keys
    2. AES private keys

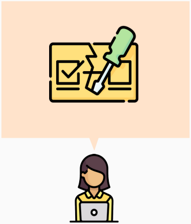
Future Directions

- More efficient instantiation with **constant-size ciphertexts**?
- Connection with **online-extractable ZK** and **commit-and-prove ZK**?
- Compiling other IOPs into VE?

Advanced Security

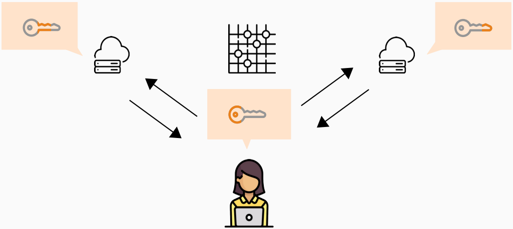


✓ What happens if the signer partially leaks randomness?



✓ What happens if the signer produces faulty signatures?

New Constructions



✓ Can we construct multi-party signatures from lattice ZK proof?

✓ Can we add verifiability to ciphertexts using ZK proof?

Publications on Advanced Security Analysis

1. *New Bleichenbacher Records: Fault Attacks on qDSA Signatures.* with Mehdi Tibouchi and Masayuki Abe. **CHES 2018**

2. *Degenerate Fault Attacks on Elliptic Curve Parameters in OpenSSL.* with Mehdi Tibouchi. **IEEE EuroS&P 2019**

3. *Security of Hedged Fiat-Shamir Signatures under Fault Attacks.* with Diego F. Aranha, Claudio Orlandi, and Greg Zaverucha. **EUROCRYPT 2020**

4. *LadderLeak: Breaking ECDSA with Less than One Bit of Nonce Leakage.* with Diego F. Aranha, Felipe Rodrigues Novaes, Mehdi Tibouchi, and Yuval Yarom. **ACM CCS 2020**, **BH Europe 2020**, and **RWC 2021**

5. *Side-channel Protections for Picnic Signatures.* with Diego F. Aranha, Sebastian Berndt, Thomas Eisenbarth, Okan Seker, Luca Wilke, and Greg Zaverucha. **CHES 2021**

6. *Fiat-Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model).* with Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, and Daniel Tschudi. **EUROCRYPT 2022**

Publications on New Cryptographic Constructions

7. *Two-round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices.* with Ivan Damgård, Claudio Orlandi, and Mehdi Tibouchi. **PKC 2021** and **JoC (Invited!)**

8. *ECLIPSE: Enhanced Compiling Method for Pedersen-committed zkSNARK Engines.* with Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, and Claudio Orlandi. **PKC 2022**

9. *MITAKA: A Simpler, Parallelizable, Maskable Variant of Falcon.* with Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. **EUROCRYPT 2022**

10. *MuSig-L: Lattice-based Multi-Signature with Single-Round Online Phase.* with Cecilia Boschini and Mehdi Tibouchi. **CRYPTO 2022**

11. *Verifiable Encryption from MPC-in-the-Head.* with Greg Zaverucha. **Under submission**

Thank you!

References i

Martin R. Albrecht and Nadia Heninger.
On bounded distance decoding with predicate: Breaking the "lattice barrier" for the hidden number problem.
In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 528–558. Springer, Heidelberg, October 2021.

Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore.
Group signatures and more from isogenies and lattices: Generic, simple, and efficient.
Cryptology ePrint Archive, Report 2021/1366, 2021.
`https://eprint.iacr.org/2021/1366`.

References ii

- Jan Camenisch and Ivan Damgård.
  Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes.
  In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer, Heidelberg, December 2000.

- Jan Camenisch and Victor Shoup.
  Practical verifiable encryption and decryption of discrete logarithms.
  In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003.

Marc Fischlin and Felix Günther.
Modeling memory faults in signature and authenticated encryption schemes.
In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 56–84. Springer, Heidelberg, February 2020.

Freepik.
Icons made by Freepik from Flaticon.com.
http://www.flaticon.com.

Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz.
Tight adaptive reprogramming in the QROM.
In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 637–667. Springer, Heidelberg, December 2021.

References iv

- Irene Giacomelli, Jesper Madsen, and Claudio Orlandi.
  ZKBoo: Faster zero-knowledge for Boolean circuits.
  In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

- Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.
  Zero-knowledge from secure multiparty computation.
  In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

References v

- Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh.
  Saver: SNARK-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization.
  Cryptology ePrint Archive, Report 2019/1270, 2019.
  https://eprint.iacr.org/2019/1270.

- Vadim Lyubashevsky and Gregory Neven.
  One-shot verifiable encryption from lattices.
  In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 293–323. Springer, Heidelberg, April / May 2017.

References vi

Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille.
MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces.
In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020.