How to do Efficient Signature Verification without Leakage

Cecilia Boschini
(joint work with Dario Fiore and Elena Pagnin)

Technion (Israel)

December, 2021

Digital Signatures

KeyGen Sign Ver

$\xrightarrow{(msg, \sigma)}$

sk $\quad\quad$ vk

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$ $\quad\quad$ $b \leftarrow \mathsf{Ver}(vk, (msg, \sigma))$

Digital Signatures

KeyGen Sign Ver

Unforgeability

(msg, σ)

(msg^*, σ^*)

sk

vk

$0 \leftarrow \mathsf{Ver}(vk, (msg^*, \sigma^*))$

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$

Digital Signatures

KeyGen Sign Ver EffVer

sk

$\xrightarrow{(msg, \sigma)}$

vk

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$

$b \leftarrow \mathsf{EffVer}(vk, (msg, \sigma))$

EffVer requires less computation than Ver $\Rightarrow$ unforgeability?

Digital Signatures

KeyGen Sign Ver EffVer

Unforgeability – Hope

sk

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$

$\xrightarrow{(msg,\sigma)}$

vk

$\xrightarrow{(msg^*,\sigma^*)}$

$0 \leftarrow \mathsf{Ver}(vk, (msg^*, \sigma^*))$

$0 \leftarrow \mathsf{EffVer}(vk, (msg^*, \sigma^*))$

Digital Signatures with Efficient Verification

KeyGen | Sign | Ver | EffVer

Unforgeability – Reality

sk

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$

(msg, σ) $\longrightarrow$

vk

(msg^*, σ^*) $\longrightarrow$

$0 \leftarrow \mathsf{Ver}(vk, (msg^*, \sigma^*))$

$1 \leftarrow \mathsf{EffVer}(vk, (msg^*, \sigma^*))$

Scheme still unforgeable, but possible to trick verifiers with small computing power!

Can we give a general framework to avoid these bugs?

Usecase: PQ ($A\sigma = v$)-style Signatures

Verify a signature σ w.r.t. $vk = (A, v)$.

Signatures
- lattices,
- multivariate equations

(1) $\|\sigma\|$ small

(2) $\underbrace{A}_{n \times m} \; \sigma = v \mod q$

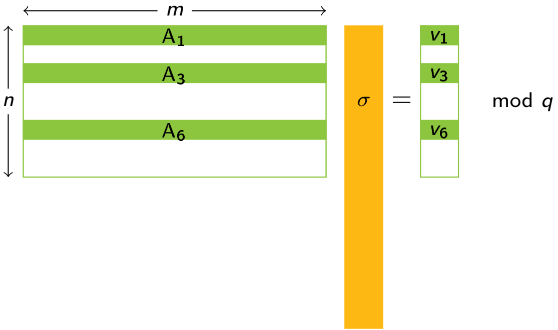
Usecase: PQ ($A\sigma = v$)-style Signatures

Faster verification: check k random rows

⚠ is the signature still unforgeable?

(1) $\|\sigma\|$ small

(2)

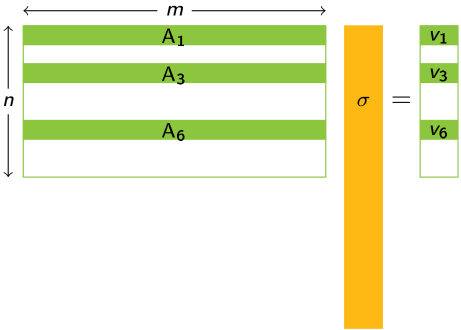


Usecase: PQ ($A\sigma = v$)-style Signatures

Faster verification: check k random rows
⚠ is the signature still unforgeable?

(1) $\|\sigma\|$ small

(2)



mod q

Assume: q is prime $\wedge$ only checked 1st row

$$a_1^1\sigma_1 + a_1^2\sigma_2 + \ldots + a_1^m\sigma_m = v_1 \mod q$$

⚠ Forgery:

- sample random (small) σ_i for $i = 2, \ldots, m$
- $\sigma_1 \leftarrow (a_1^1)^{-1} \cdot (v_1 - a_1^2\sigma_2 + \ldots + a_1^m\sigma_m) \mod q$
  (and hope it's small)

Usecase: PQ ($A\sigma = v$)-style Signatures

Faster verification: check k random **linear combinations** of the rows of A

⚠ is the signature still unforgeable?

(1) $\|\sigma\|$ small

(2) $\underbrace{C \cdot A}_{k \times m} \, \sigma = Cv \mod q$ for randomly chosen $C \in \mathbb{Z}_q^{k \times n}$

(1) is this secure?
(2) can we use the same C multiple times?
⚠ LEAKAGE!

Our results

Efficiency

We present a general framework to analyze the security of a signature scheme Σ augmented with efficient verification.

Concrete Security for signatures with $(A\sigma = v)$-style verification

Let Σ be an existentially unforgeable signature scheme with $(A\sigma = v)$-style verification. The scheme $\Sigma^E = (\Sigma, \mathrm{EffVer})$ is existentially unforgeable under adaptive chosen message and verification attacks. Concretely, the advantage of the adversary is bounded by $\frac{q_V+1}{q^k-q_V}$.

Leakage

Each verification query leaks an amount of information about C proportional to $\approx \frac{1}{q^k}$.

Flexibility

The framework can be extended to securely extract information from interrupted verification.

Talk Overview

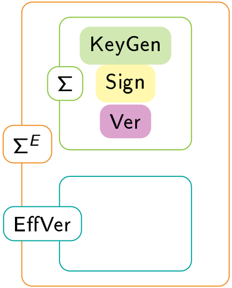
Formal Model of Efficient Verification

Application to PQ signatures

Future Directions

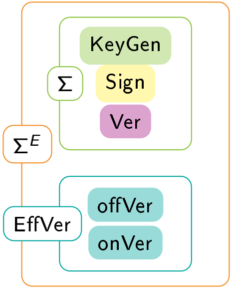
Signature with Efficient Verification

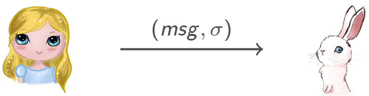
Extra algorithm, more efficient but lower security (otherwise it is just a better version of Σ).



Signature with Efficient Verification

Extra algorithm, more efficient but lower security (otherwise it is just a better version of Σ).





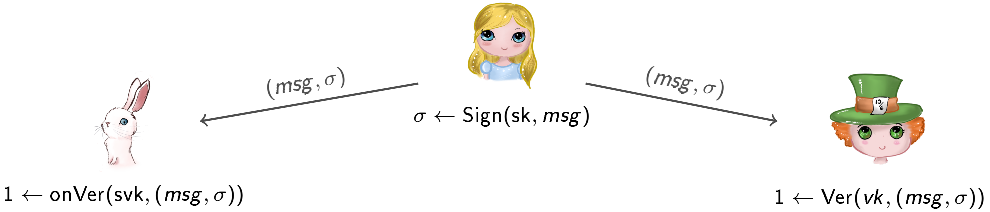
sk

$\mathsf{svk} \leftarrow \mathsf{offVer}(vk, k)$

$b \leftarrow \mathsf{onVer}(\mathsf{svk}, (msg, \sigma))$

- offline/online
- confidence level $k \in \{1, \dots, n\}$ (set by verifier; e.g., number of rows it can check)
- secret verification key svk (derived from k)

Correctness



Correctness:
Honestly generated signatures are always accepted:

$$\Pr\left(\mathsf{onVer}(\mathsf{svk}, (msg, \sigma)) = 1 \mid \mathsf{Ver}(vk, (msg, \sigma)) = 1\right) = 1$$

(on top of correctness of Σ).

Efficiency

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)$

$(msg_1, \sigma_1), \ldots, (msg_r, \sigma_r)$

$\mathsf{svk} \leftarrow \mathsf{offVer}(vk, k)$

For $i = 1, \ldots, \mathsf{r}$:

$1 \leftarrow \mathsf{onVer}(\mathsf{svk}, (msg_i, \sigma_i))$

$(msg_1, \sigma_1), \ldots, (msg_r, \sigma_r)$

For $i = 1, \ldots, \mathsf{r}$:

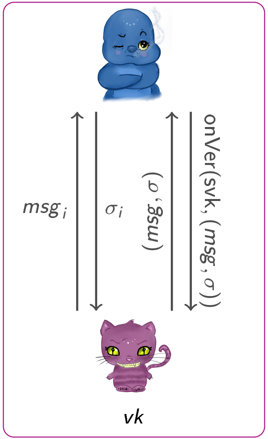
$1 \leftarrow \mathsf{Ver}(vk, (msg_i, \sigma_i))$

$(\mathsf{r}_0, \mathsf{e}_0)$-Concrete Amortized Efficiency:

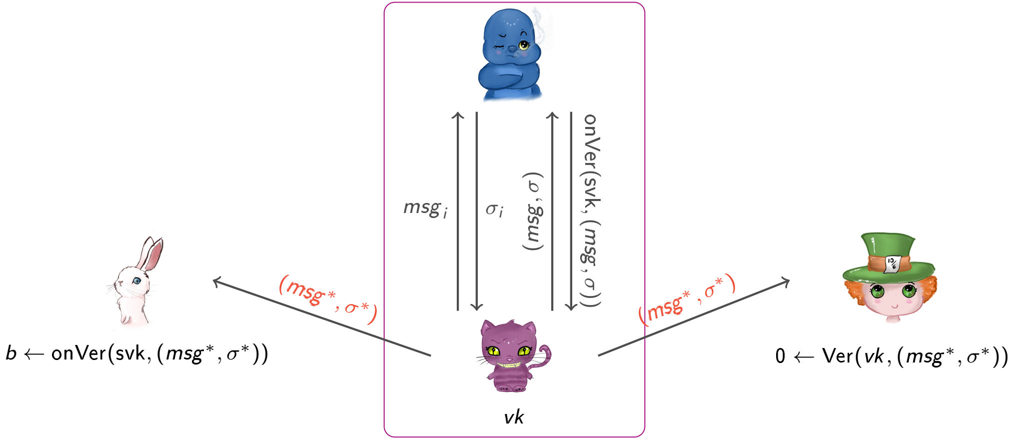
When verifying many signatures, onVer is much faster than Ver:

$$\forall\ \mathsf{r} \geq \mathsf{r}_0\ , \quad \frac{\mathsf{cost}\big(\mathsf{offVer} + \mathsf{r} \cdot \mathsf{onVer}\big)}{\mathsf{cost}\big(\mathsf{r} \cdot \mathsf{Ver}\big)} < \mathsf{e}_0\ .$$

Unforgeability = Hard to find False Positives



Unforgeability = Hard to find False Positives



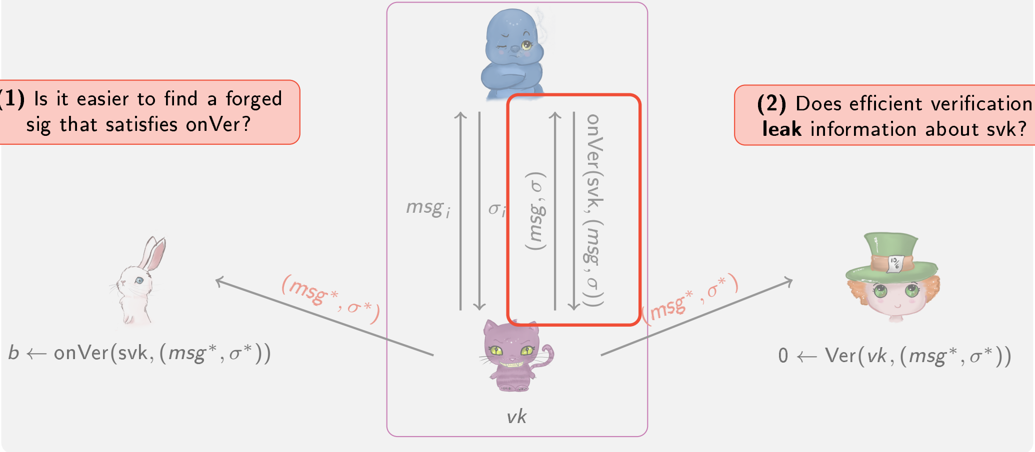
Unforgeability:

Fixed svk, given $\mathcal{A}$ oracle access to onVer and Sign it holds $\Pr(b = 1) = \varepsilon$ (on top of unforgeability of Σ).

Unforgeability $=$ Hard to find False Positives

(1) Is it easier to find a forged sig that satisfies onVer?

(2) Does efficient verification **leak** information about svk?



Unforgeability:

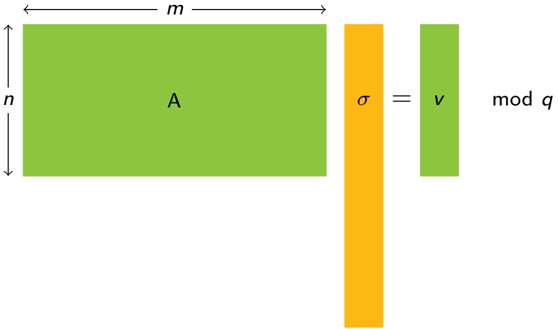
Fixed svk, given $\mathcal{A}$ oracle access to onVer and Sign it holds $\Pr(b = 1) = \varepsilon$ (on top of unforgeability of Σ).

Ver for PQsig: Hash-and-Sign from Lattices

Verify a signature σ w.r.t. $vk = (\mathsf{A}, v = \mathcal{H}(msg))$.

(1) $\|\sigma\|$ small

(2)



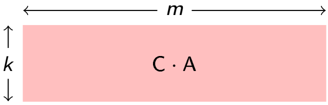

```
Ver(vk, (msg, σ))
// INITIALIZE ACCEPTANCE BITS
    b1 ← 0, b2 ← 0
// SPLIT vk INTO MATRIX - AUX. DATA
    parse vk = (PK, PK.aux)
// ADDITIONAL VERIFICATION CHECKS
    b1 ← Check(PK.aux, msg, σ)
// FORMATTING (Aσ = v)-STYLE CHECK
    v ← H(msg)
// MATRIX-VECTOR MULT. CHECK
    if (A · σ = v)
        b2 ← 1
    return (b1 ∧ b2)
```

[STOC:GPV08] Craig Gentry, Chris Peikert, Vinod Vaikuntanathan "Trapdoors for hard lattices and new cryptographic constructions", STOC 2008.

EffVer for PQsig: Hash-and-Sign from Lattices

Verify a signature σ w.r.t. $vk = (\mathsf{A}, \mathcal{H}(msg))$.

(2)




```
offVer(vk, k)
----------------------------------------------
// CHECK PARAMETER CONSISTENCY
1 :    if (k > n ∨ k < 1)   return ⊥
// GENERATE RANDOMIZED KEY
2 :    Z ← GetZ(A, k)
   i :    for j = 1, ..., k
   ii :      c ←$ Z_q^{1×n}
   iii :     z ← c[A | −I_{n×n}] ∈ Z_q^{1×(m+1)}
   iv :      if z ∈ ⟨z_0, ..., z_{j−1}⟩_q
             go to ii.
   v :       z_j ← z
   vi :   set Z ← [z_1^T | ... | z_k^T]^T
3 :    return svk ← (k, Z, PK.aux)
```

[STOC:GPV08] Craig Gentry, Chris Peikert, Vinod Vaikuntanathan "Trapdoors for hard lattices and new cryptographic constructions", STOC 2008.

EffVer for PQsig: Hash-and-Sign from Lattices

Verify a signature σ w.r.t. $vk = (\mathsf{A}, \mathcal{H}(msg))$.

(1) $\|\sigma\|$ small

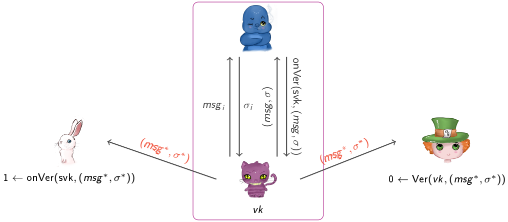
(2) $\underbrace{\mathsf{C} \cdot \mathsf{A}}_{k \times m} \, \sigma = \mathsf{C}v \mod q$

$\mathsf{onVer}(\mathsf{svk}, msg, \sigma)$

```
// LIGHTWEIGHT VERIFICATION CHECKS
if Check(PK.aux, msg, σ) = 0
    return 0
// FORMATTING FOR EFFICIENT VERIF.
Z' ← [CA | −C]
σ' ← [σ ; H(msg)]
parse Z' = [z'_1^T | ... | z'_k^T]^T ∈ Z_q^{k×cols(Z')}
// LINE-BY-LINE INNER PRODUCTS
for j = 1, ..., k
    if z'_j · σ' ≠ 0 mod q
        return 0
return 1
```

[STOC:GPV08] Craig Gentry, Chris Peikert, Vinod Vaikuntanathan "Trapdoors for hard lattices and new cryptographic constructions", STOC 2008.

Unforgeability proof



$$\text{forgery} = \underbrace{(\text{forgery against } \mathsf{Ver})}_{\text{impossible if } \Sigma \text{ unforgeable}} + \underbrace{(\text{forgery against } \mathsf{onVer})}_{\mathsf{bad}}$$

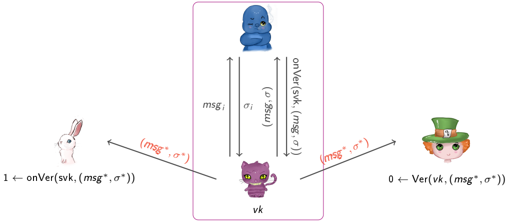
1. $\mathcal{A}$ wins in the event $\mathsf{bad} := \{\exists\ i \in \{1, \ldots, q_V + 1\} : \mathsf{Ver}(vk, msg_i, \sigma_i) = 0 \ \wedge \ \mathsf{onVer}(\mathsf{svk}, msg_i, \sigma_i) = 1\}$.

2. The adversary could try submitting a forgery whenever it queries the verification oracle:

$$\Pr[\mathsf{bad}] \leq \sum_{i=1}^{q_V+1} \Pr\left[\mathsf{bad}_i \middle| \bigwedge_{j=1}^{i-1} \neg \mathsf{bad}_j\right]$$

3. Analyze the leakage in the (rejected) verification queries.

Unforgeability proof



$$\text{forgery} = \underbrace{(\text{forgery against } \mathsf{Ver})}_{\text{impossible if } \Sigma \text{ unforgeable}} + \underbrace{(\text{forgery against } \mathsf{onVer})}_{\mathsf{bad}}$$

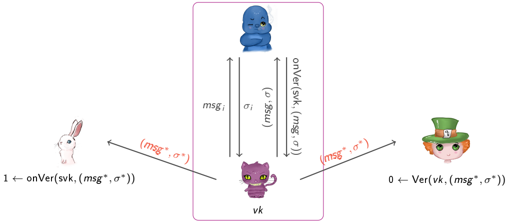
1. $\mathcal{A}$ wins in the event bad $:= \{\exists\ i \in \{1, \ldots, q_V + 1\} : \mathsf{Ver}(vk, msg_i, \sigma_i) = 0 \ \wedge \ \mathsf{onVer}(\mathsf{svk}, msg_i, \sigma_i) = 1\}$.

2. The adversary could try submitting a forgery whenever it queries the verification oracle:

$$\Pr[\mathsf{bad}] \leq \sum_{i=1}^{q_V+1} \Pr\left[\mathsf{bad}_i | \bigwedge_{j=1}^{i-1} \neg \mathsf{bad}_j\right]$$

3. Analyze the leakage in the (rejected) verification queries.

Unforgeability proof



$$\text{forgery} = \underbrace{(\text{forgery against } \mathsf{Ver})}_{\text{impossible if } \Sigma \text{ unforgeable}} + \underbrace{(\text{forgery against } \mathsf{onVer})}_{\mathsf{bad}}$$

1. $\mathcal{A}$ wins in the event bad $:= \{\exists\ i \in \{1, \dots, q_V + 1\} : \mathsf{Ver}(vk, msg_i, \sigma_i) = 0 \ \wedge\ \mathsf{onVer}(\mathsf{svk}, msg_i, \sigma_i) = 1\}$.

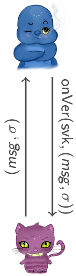
2. The adversary could try submitting a forgery whenever it queries the verification oracle:

$$\Pr[\mathsf{bad}] \leq \sum_{i=1}^{q_V+1} \underbrace{\Pr\left[\mathsf{bad}_i \middle| \bigwedge_{j=1}^{i-1} \neg\mathsf{bad}_j\right]}_{\leq \frac{1}{q^k-(i-1)}} = \frac{q_V + 1}{q^k - q_V}$$

3. Analyze the leakage in the (rejected) verification queries.

Verification Leakage

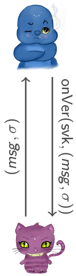
Knowing $w := \underbrace{A\sigma - v \mod q}_{\text{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{Cw \mod q}_{\text{from onVer}}$?



Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

Verification Leakage

Knowing $\mathsf{w} := \underbrace{\mathsf{A}\sigma - v \mod q}_{\textbf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathsf{Cw} \mod q}_{\textbf{from onVer}}$?

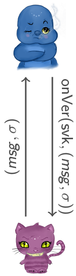


1. no info: $\mathsf{w} = 0 \mod q \quad \Rightarrow \quad \mathsf{Cw} = 0 \mod q$
2. impossible by Correctness
3. bad
4. $\mathsf{Cw} \neq 0 \mod q \ \wedge\ \mathsf{w} \neq 0 \mod q$
   ⚠ LEAKAGE: the rows of C are not in the left kernel of w!

Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

Verification Leakage

Knowing $\underbrace{\mathsf{w} := \mathsf{A}\sigma - v \mod q}_{\text{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathsf{C}\mathsf{w} \mod q}_{\text{from onVer}}$?

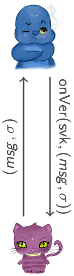


1. no info: $\mathsf{w} = 0 \mod q \quad \Rightarrow \quad \mathsf{C}\mathsf{w} = 0 \mod q$
2. impossible by Correctness
3. bad
4. $\mathsf{C}\mathsf{w} \neq 0 \mod q \ \wedge \ \mathsf{w} \neq 0 \mod q$
   ⚠ LEAKAGE: the rows of C are not in the left kernel of w!

Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

Verification Leakage

Knowing $\underbrace{\mathsf{w} := \mathsf{A}\sigma - v \mod q}_{\textbf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathsf{Cw} \mod q}_{\textbf{from onVer}}$?

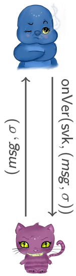


Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

1. no info: $\mathsf{w} = 0 \mod q \quad \Rightarrow \quad \mathsf{Cw} = 0 \mod q$
2. impossible by Correctness
3. bad
4. $\mathsf{Cw} \neq 0 \mod q \ \wedge \ \mathsf{w} \neq 0 \mod q$
   ⚠ LEAKAGE: the rows of C are not in the left kernel of w!

Verification Leakage

Knowing $\underbrace{\mathrm{w} := \mathrm{A}\sigma - v \mod q}_{\text{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathrm{Cw} \mod q}_{\text{from onVer}}$?

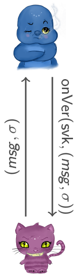


1. no info: $\mathrm{w} = 0 \mod q \quad \Rightarrow \quad \mathrm{Cw} = 0 \mod q$
2. impossible by Correctness
3. bad
4. $\mathrm{Cw} \neq 0 \mod q \ \wedge \ \mathrm{w} \neq 0 \mod q$
   ⚠ LEAKAGE: the rows of C are not in the left kernel of w!

Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

Verification Leakage

Knowing $\mathrm{w} := \underbrace{\mathrm{A}\sigma - v \mod q}_{\textsf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathrm{C}\mathrm{w} \mod q}_{\textsf{from onVer}}$?



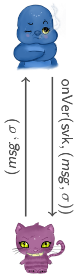
Before querying phase

$$C_0 := |\mathcal{C}| = \left|\left\{\mathrm{C} \in \mathbb{Z}_q^{k \times m} \mid rk(\mathrm{C}) = k\right\}\right|$$

Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	**0**	**0**

Verification Leakage

Knowing $\mathsf{w} := \underbrace{\mathsf{A}\sigma - v \mod q}_{\textbf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathsf{Cw} \mod q}_{\textbf{from onVer}}$?



Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	**0**	**0**

Before querying phase

$$C_0 := |\mathcal{C}| = \left|\left\{\mathsf{C} \in \mathbb{Z}_q^{k\times m} \mid rk(\mathsf{C}) = k\right\}\right|$$

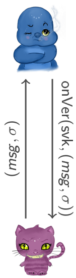
Leakage in i-th Rejected Verification Query

$$C_i = \left|\mathcal{C} \setminus \cup_{j=1}^{i-1} \mathcal{H}_j\right|$$

where $\mathcal{H}_i := \{\text{matrices whose rows are in the left kernel of } (\mathsf{A}_j\sigma_j - v_j)\}$.

Verification Leakage

Knowing $\underbrace{\mathrm{w} := \mathrm{A}\sigma - v \mod q}_{\textbf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathrm{Cw} \mod q}_{\textbf{from onVer}}$?



Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	**0**	**0**

Before querying phase

$$C_0 := |\mathcal{C}| = \left|\left\{\mathrm{C} \in \mathbb{Z}_q^{k\times m} \mid rk(\mathrm{C}) = k\right\}\right|$$

Leakage in i-th Rejected Verification Query

$$C_i = \left|\mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right|$$

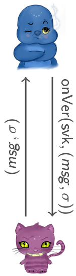
where $\mathcal{H}_i := \{\text{matrices whose rows are in the left kernel of } (\mathrm{A}_j\sigma_j - v_j)\}$.

Probability of bad_i

$$\Pr\left[\mathsf{bad}_i \mid \bigwedge_{j=1}^{i-1} \neg\mathsf{bad}_j\right] \leq \frac{\Pr\left[\mathrm{Cw}_i = 0 \mid \mathrm{C} \xleftarrow{\$} \mathcal{C}\right]}{\Pr\left[\mathrm{C} \xleftarrow{\$} \mathcal{C} \wedge \mathrm{C} \notin \mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right]} = \frac{|\mathcal{H}_1|}{\left|\mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right|}$$

Verification Leakage

Knowing $\mathrm{w} := \underbrace{\mathrm{A}\sigma - v \mod q}_{\textbf{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathrm{Cw} \mod q}_{\textbf{from onVer}}$?



Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	0	0

Before querying phase

$$C_0 := |\mathcal{C}| = \left|\left\{\mathrm{C} \in \mathbb{Z}_q^{k\times m} \mid rk(\mathrm{C}) = k\right\}\right|$$

Leakage in i-th Rejected Verification Query

$$C_i = \left|\mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right| \geq |\mathcal{H}_1| \left(\frac{q^n - 1}{q^{n-k} - 1} - (i-1)\right)$$

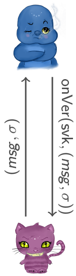
where $\mathcal{H}_i := \{\text{matrices whose rows are in the left kernel of } (\mathrm{A}_j\sigma_j - v_j)\}$.

Probability of bad_i

$$\Pr\left[\mathsf{bad}_i \mid \bigwedge_{j=1}^{i-1} \neg\mathsf{bad}_j\right] \leq \frac{\Pr\left[\mathrm{Cw}_i = 0 \mid \mathrm{C} \xleftarrow{\$} \mathcal{C}\right]}{\Pr\left[\mathrm{C} \xleftarrow{\$} \mathcal{C} \wedge \mathrm{C} \notin \mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right]} = \frac{|\mathcal{H}_1|}{\left|\mathcal{C} \setminus \cup_{j=1}^{i-1}\mathcal{H}_j\right|}$$

Verification Leakage

Knowing $\mathsf{w} := \underbrace{\mathsf{A}\sigma - v \mod q}_{\text{from Ver}}$ how much information can $\mathcal{A}$ extract from $\underbrace{\mathsf{Cw} \mod q}_{\text{from onVer}}$?



Accept		Reject	
onVer	Ver	onVer	Ver
1	1	0	1
1	0	**0**	**0**

Before querying phase

$$C_0 := |\mathcal{C}| = \left|\left\{\mathsf{C} \in \mathbb{Z}_q^{k \times m} \mid rk(\mathsf{C}) = k\right\}\right|$$

Leakage in i-th Rejected Verification Query

$$C_i = \left|\mathcal{C} \setminus \cup_{j=1}^{i-1} \mathcal{H}_j\right| \geq |\mathcal{H}_1| \left(\frac{q^n - 1}{q^{n-k} - 1} - (i-1)\right)$$

where $\mathcal{H}_i := \{\text{matrices whose rows are in the left kernel of } (\mathsf{A}_j\sigma_j - v_j)\}$.

Probability of bad_i

$$\Pr\left[\mathsf{bad}_i \mid \bigwedge_{j=1}^{i-1} \neg\mathsf{bad}_j\right] \leq \frac{|\mathcal{H}_1|}{|\mathcal{H}_1| \cdot \left(\frac{q^n-1}{q^{n-k}-1} - (i-1)\right)} \leq \frac{1}{q^k - (i-1)}$$

Speedup Estimates

Ring or Field Size	Min. Accuracy Level for 128-bit security	Concrete Amortized Efficiency	Online Efficiency $\frac{\text{cost(onVer)}}{\text{cost(Ver)}} = \frac{k_0}{n}$
exponential: $q = 2^{128}$ [AC:FMNP16]; [STOC:GVW15]	$k_0 = 1$	$(r_0 = 2, e_0 = 0.51)$	$\frac{1}{256} < 0.4\%$
mid-size poly.: $q = 2^{26}$ [EC:Lyu12]	$k_0 = 7$	$(r_0 = 8, e_0 = 0.89)$	$\frac{7}{512} < 1.4\%$
small poly.: $q = 16$ $\mathbb{F}_{2^4}$-$(32, 32, 32)$ [ACNS:Rainbow05]	$k_0 = 32$	$(r_0 = 65, e_0 = 0.99)$	$\frac{32}{64} = 50\%$

$q_V = 2^{30}$

[AC:FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, Elena Pagnin "Multi-key Homomorphic Authenticators", ASIACRYPT 2016.
[STOC:GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, Daniel Wichs "Leveled Fully Homomorphic Signatures from Standard Lattices", STOC 2015.
[EC:Lyu12] Vadim Lyubashevsky "Lattice Signatures without Trapdoors", EUROCRYPT 2012.
[ACNS:Rainbow05] Jintai Ding, Dieter Schmidt "Rainbow, a New Multivariable Polynomial Signature Scheme", ACNS 2005.

Batch Verification?

Faster verification: check k signatures σ_i on msg_i at the same time

⚠ is the signature still unforgeable?

(1) $\|\sigma_i\|$ small for $i = 1, \dots, k$

(2) $$\underbrace{A}_{n \times m} \left(\sigma_1 + \cdots + \sigma_k \right) = \left(v_1 + \cdots + v_k \right) \bmod q$$

⚠ This does not work for all the ($A\sigma = v$)-style signatures, as A could depend on *msg* too!

Many signers (Lattice-based Signatures)

How to verify N signatures from different signers on the same message?

$$\mathsf{A}_j \sigma_j = v_j \mod q \text{ for } j = 1, \ldots, N$$

Schnorr ID scheme

$\mathcal{P}$ (s)		$\mathcal{V}$ (A, v)
y $\mathsf{t} = \mathsf{Ay} \mod q$		
	$\xrightarrow{\mathsf{t}}$	
		$c \xleftarrow{\$} \mathcal{C}$
	$\xleftarrow{c}$	
$\mathsf{z} = \mathsf{y} + c\mathsf{s} \mod q$		
	$\xrightarrow{\mathsf{z}}$	
		$\lVert\mathsf{z}\rVert$ *small?* $\mathsf{Az} \overset{?}{\equiv_q} \mathsf{t} + cv$

Multisignature

A valid signature on *msg* has to be generated by all N signers.

Many signers (Lattice-based Signatures)

How to verify N signatures from different signers on the same message?

$$\mathsf{A}_j \sigma_j = v_j \mod q \text{ for } j = 1, \ldots, N$$

Schnorr ID scheme

$\mathcal{P}$ (s)		$\mathcal{V}$ (A, v)
$\mathsf{y} \xleftarrow{\$} \mathcal{D}$ $\mathsf{t} = \mathsf{Ay} \mod q$		
	$\xrightarrow{\mathsf{t}}$	
		$c \xleftarrow{\$} \mathcal{C}$
	$\xleftarrow{c}$	
$\mathsf{z} = \mathsf{y} + c\mathsf{s} \mod q$ reject.sampl.		
	$\xrightarrow{\mathsf{z}}$	
		$\|\mathsf{z}\|$ *small?* $\mathsf{Az} \overset{?}{\equiv}_q \mathsf{t} + cv$

Multisignature

A valid signature on *msg* has to be generated by all N signers.

Many signers (Lattice-based Signatures)

How to verify N signatures from different signers on the same message?

$$A_j\sigma_j = v_j \mod q \text{ for } j = 1, \dots, N$$

Schnorr ID scheme + Fiat-Shamir

$\mathcal{P}$ (s)		$\mathcal{V}$ (A, v)
$y \xleftarrow{\$} \mathcal{D}$ $t = Ay \mod q$		
$c = \mathcal{H}(A, v, t, msg)$		
$z = y + cs \mod q$ reject.sampl.		
	$\xrightarrow{z,c,t}$	
		$\lVert z \rVert$ *small?* $Az \overset{?}{\equiv}_q t + cv$

Signature: $\sigma = (t, c, z)$.

Multisignature

A valid signature on *msg* has to be generated by all N signers.

Many signers (Lattice-based Signatures)

How to verify N signatures from different signers on the same message?

$$\mathrm{A}_j\sigma_j = v_j \mod q \text{ for } j = 1, \ldots, N$$

Schnorr ID scheme + Fiat-Shamir

$\mathcal{P}$ (s)		$\mathcal{V}$ (A, v)
$\mathrm{y} \xleftarrow{\$} \mathcal{D}$ $\mathrm{t} = \mathrm{Ay} \mod q$		
$c = \mathcal{H}(\mathrm{A}, v, \mathrm{t}, msg)$		
$\mathrm{z} = \mathrm{y} + c\mathrm{s} \mod q$ reject.sampl.		
	$\xrightarrow{\mathrm{z},c,\mathrm{t}}$	
		$\|\mathrm{z}\|$ *small?* $\mathrm{Az} \overset{?}{\equiv}_q \mathrm{t} + cv$

Signature: $\sigma = (\mathrm{t}, c, \mathrm{z})$.

Multisignature

A valid signature on *msg* has to be generated by all N signers.

$(t, N)-$threshold Signature

A valid signature on *msg* has to be generated by at least t out of the possible t signers.

Conclusions

- Framework for signatures with efficient verification.
- Application to PQ signatures.
- Extension to the case of flexible verification (not in the talk)
  (can one extract information from a verification algorithm that was interrupted in medias res?)
- General model to deal with both efficient and flexible verification (not in the talk).

Open Questions

1. Can the model be applied to analyze the verification of other primitives (e.g., ZK proofs)?
2. Can we use this approach to do distributed verification of signatures?
   IDEA: distribute verification of a signature among N different verifiers. Unforgeability is preserved as long as t out of the N verifiers are honest.
3. how to build PQ multisignatures/threshold signatures?

Conclusions

- Framework for signatures with efficient verification.
- Application to PQ signatures.
- Extension to the case of flexible verification (not in the talk)
  (can one extract information from a verification algorithm that was interrupted in medias res?)
- General model to deal with both efficient and flexible verification (not in the talk).

Open Questions

1. Can the model be applied to analyze the verification of other primitives (e.g., ZK proofs)?
2. Can we use this approach to do distributed verification of signatures?
   IDEA: distribute verification of a signature among N different verifiers. Unforgeability is preserved as long as t out of the N verifiers are honest.
3. how to build PQ multisignatures/threshold signatures?

Thanks for listening!

Full version: `www.eprint.org/2021/832`

Drawings by Chiara Boschini.