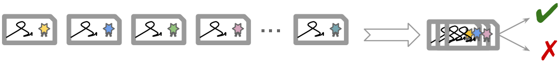
Cover-free families, constructions and cryptographical applications

Lucia Moura
University of Ottawa

Thais Bardini Idalino
Universidade Federal de Santa Catarina

Carleton Combinatorics Meeting, August 4th 2021

- Cryptography problems with a "all or nothing" solution.



- Cryptography problems with a “all or nothing” solution.

- Cover-free families to provide fault-tolerance.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg1:	1	1	1	0	0	0
agg2:	1	0	0	1	1	0
agg3:	0	1	0	1	0	1
agg4:	0	0	1	0	1	1

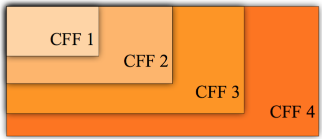
- Cryptography problems with a “all or nothing” solution.

- Cover-free families to provide fault-tolerance.

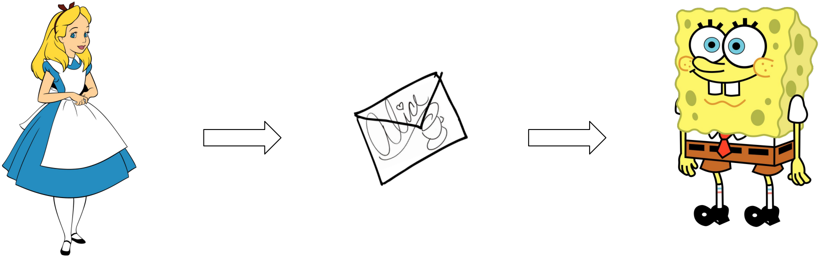
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg1:	1	1	1	0	0	0
agg2:	1	0	0	1	1	0
agg3:	0	1	0	1	0	1
agg4:	0	0	1	0	1	1

- Explore different aspects of cover-free families.

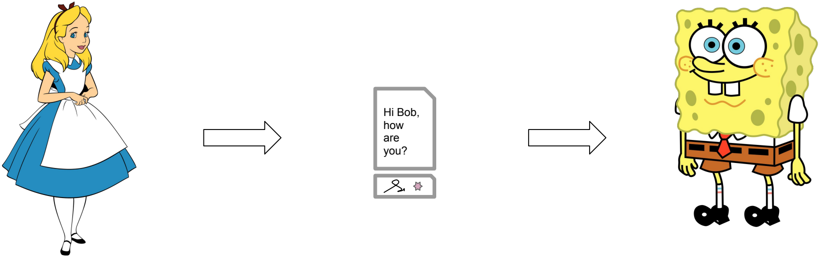
$\sigma_1 \ldots \sigma_{n_1} \ldots \sigma_{n_2} \ldots \sigma_{n_3} \ldots \sigma_{n_4}$

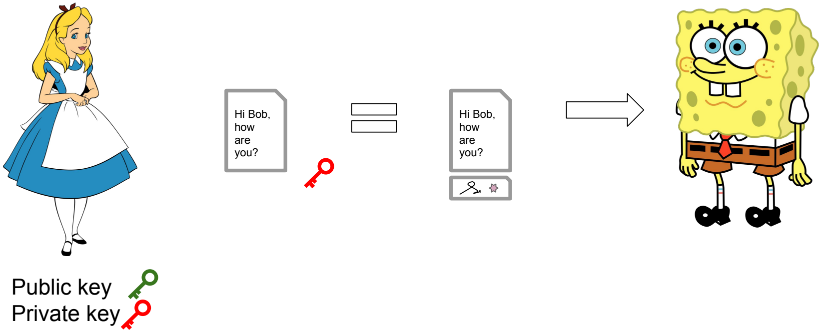


Digital signatures

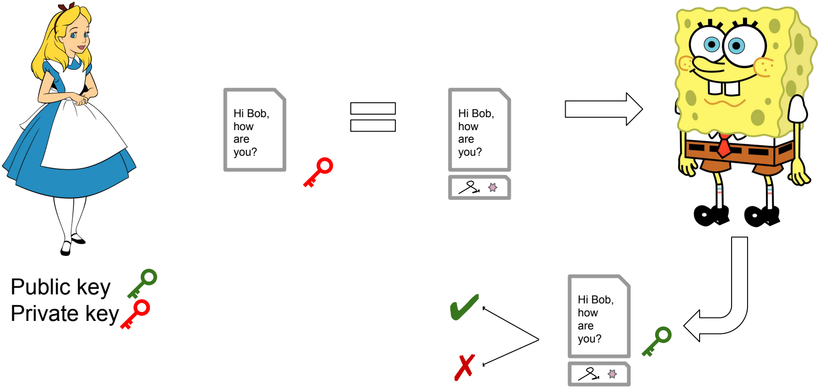


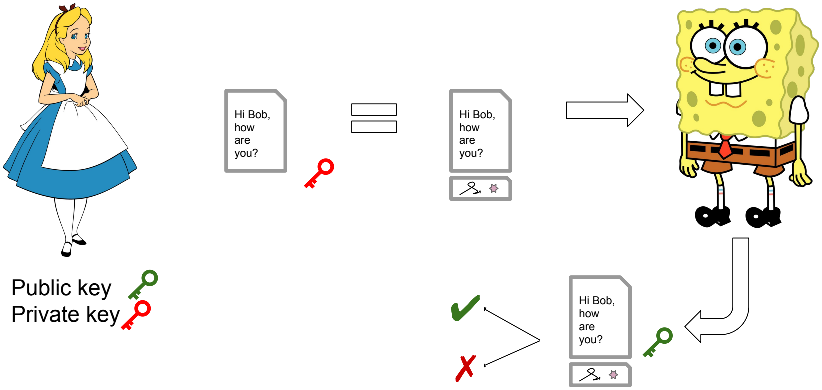
Digital signatures




Hi Bob,
how
are
you?
Hi Bob,
how
are
you?
Public key
Private key

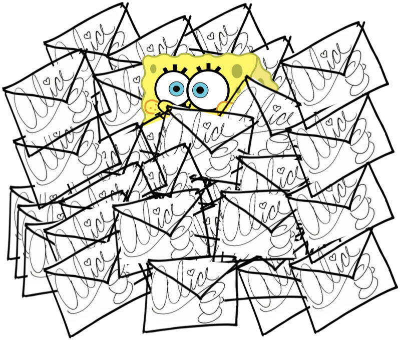
Digital signatures





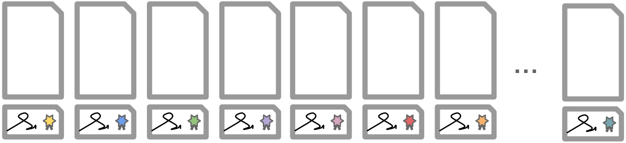
- Allows Bob to verify that the message was not modified during transmission (**integrity**), and that Alice in fact signed it (**authenticity**).

What happens when we have thousands of messages and signatures?



Aggregation of signatures

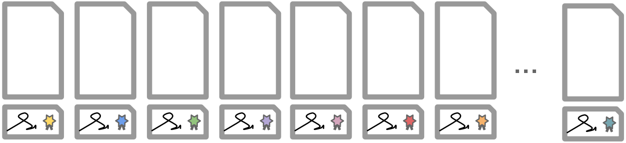
- What happens when we have thousands of msgs/signatures?



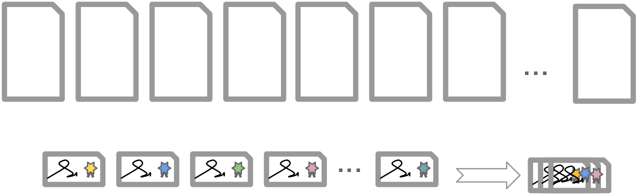
[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Eurocrypt 2003.

Aggregation of signatures

- What happens when we have thousands of msgs/signatures?



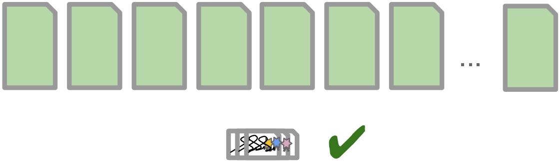
- *Aggregation of signatures*, Boneh et al. (2003)[1].



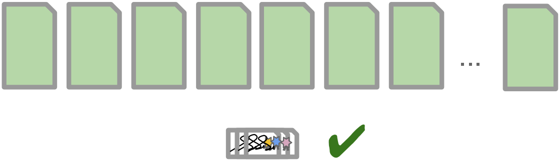
[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Eurocrypt 2003.

Aggregation of signatures

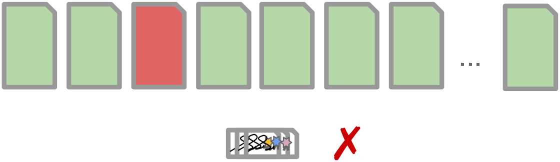
- Saves on storage, communication and verification time.



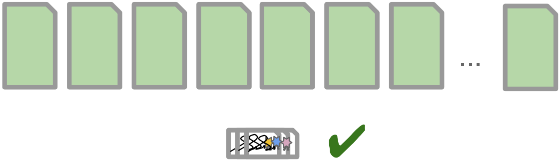
- Saves on storage, communication and verification time.



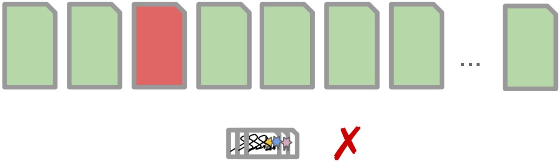
- One invalid signature invalidates the entire aggregate.



- Saves on storage, communication and verification time.

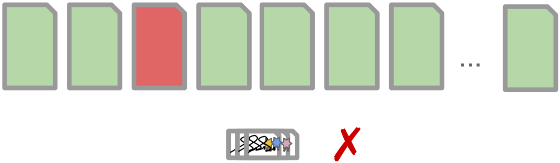


- One invalid signature invalidates the entire aggregate.



- Use d-cover-free families to provide fault-tolerance.

- One invalid signature invalidates the entire aggregate.



- Use *d*-cover-free families to provide fault-tolerance.
  - G. Zaverucha, D. Stinson, ICITS 2009.
  - T. B. Idalino. Using combinatorial group testing to solve integrity issues. Master's thesis, 2015.
  - G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

A $t \times n$ binary incidence matrix.

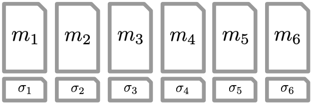
- $n =$ number of elements to be tested
- $d =$ max number of invalid elements.

	1	2	3	4	5	6	Test result:
test 1:	1	1	1	0	0	0	X
test 2:	1	0	0	1	1	0	✓
test 3:	0	1	0	1	0	1	✓
test 4:	0	0	1	0	1	1	X

1-CFF($t = 4, n = 6$)

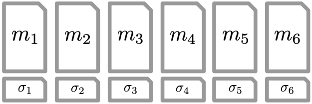
Fault tolerance with d-CFFs

- n = number of signatures
- d = max number of invalid signatures.



Fault tolerance with d-CFFs

- n = number of signatures
- d = max number of invalid signatures.



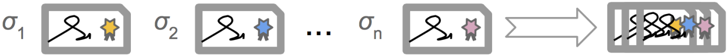
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	
agg 1:	1	1	1	0	0	0	$\sigma^*[1] = \mathbf{Agg}(\sigma_1, \sigma_2, \sigma_3)$
agg 2:	1	0	0	1	1	0	$\sigma^*[2] = \mathbf{Agg}(\sigma_1, \sigma_4, \sigma_5)$
agg 3:	0	1	0	1	0	1	$\sigma^*[3] = \mathbf{Agg}(\sigma_2, \sigma_4, \sigma_6)$
agg 4:	0	0	1	0	1	1	$\sigma^*[4] = \mathbf{Agg}(\sigma_3, \sigma_5, \sigma_6)$

$$\textbf{AggVerify}(\sigma^*[1], m_1, m_2, m_3) \ \text{✗}$$
$$\textbf{AggVerify}(\sigma^*[2], m_1, m_4, m_5) \ \checkmark$$
$$\textbf{AggVerify}(\sigma^*[3], m_2, m_4, m_6) \ \checkmark$$
$$\textbf{AggVerify}(\sigma^*[4], m_3, m_5, m_6) \ \text{✗}$$

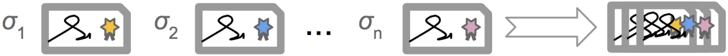
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	result:
agg 1:	1	1	1	0	0	0	✗
agg 2:	1	0	0	1	1	0	✓
agg 3:	0	1	0	1	0	1	✓
agg 4:	0	0	1	0	1	1	✗

Invalid signature: σ_3

- **Before**: dynamically aggregate signatures as they arrive.



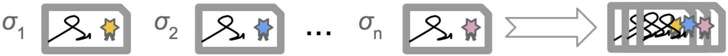
- **Before**: dynamically aggregate signatures as they arrive.



- **Now**: the number of signatures is bounded by n.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg 1:	1	1	1	0	0	0
agg 2:	1	0	0	1	1	0
agg 3:	0	1	0	1	0	1
agg 4:	0	0	1	0	1	1

- **Before**: dynamically aggregate signatures as they arrive.



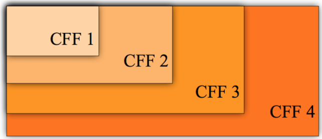
- **Now**: the number of signatures is bounded by n.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg 1:	1	1	1	0	0	0
agg 2:	1	0	0	1	1	0
agg 3:	0	1	0	1	0	1
agg 4:	0	0	1	0	1	1

- Impractical for applications where signatures are dynamically arriving.

How to make the number of signatures dynamic and still guarantee a reasonable size for the aggregate signature?

$\sigma_1 \ldots \sigma_{n_1} \ldots \sigma_{n_2} \ldots \sigma_{n_3} \ldots \sigma_{n_4}$



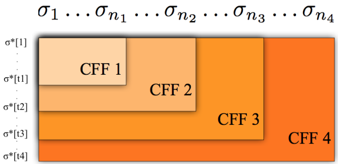
- **Problem:** Fault-tolerant aggregation of signatures with unknown n.

- **Problem:** Fault-tolerant aggregation of signatures with unknown n.
- **Solution:** Increase the d-CFF to hold extra signatures.

- **Problem:** Fault-tolerant aggregation of signatures with unknown n.
- **Solution:** Increase the d-CFF to hold extra signatures.
- Create a special sequence of d-CFF matrices.

1-CFF(5,10) Matrix

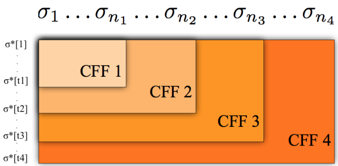
	1	2	3	4	5	6	7	8	9	10
$test_1$	1	1	1	0	0	0	1	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0
$test_3$	0	1	0	1	0	1	0	0	1	0
$test_4$	0	0	1	0	1	1	0	0	0	1
$test_5$	0	0	0	0	0	0	1	1	1	1



- **Problem:** Fault-tolerant aggregation of signatures with unknown n.
- **Solution:** Increase the d-CFF to hold extra signatures.
- Create a special sequence of d-CFF matrices.
    - Large matrices contain small matrices.

1-CFF(5,10) Matrix

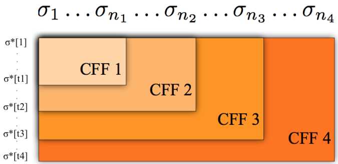
	1	2	3	4	5	6	7	8	9	10
$test_1$	1	1	1	0	0	0	1	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0
$test_3$	0	1	0	1	0	1	0	0	1	0
$test_4$	0	0	1	0	1	1	0	0	0	1
$test_5$	0	0	0	0	0	0	1	1	1	1



- **Problem:** Fault-tolerant aggregation of signatures with unknown n.
- **Solution:** Increase the d-CFF to hold extra signatures.
- Create a special sequence of d-CFF matrices.
    - Large matrices contain small matrices.
    - Avoid using unavailable signatures in the new aggregates.

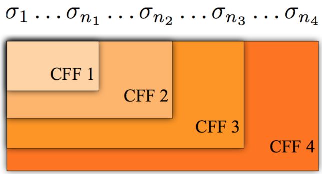
1-CFF(5,10) Matrix

	1	2	3	4	5	6	7	8	9	10
$test_1$	1	1	1	0	0	0	1	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0
$test_3$	0	1	0	1	0	1	0	0	1	0
$test_4$	0	0	1	0	1	1	0	0	0	1
$test_5$	0	0	0	0	0	0	1	1	1	1



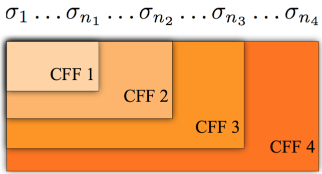
Compression Ratio

- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$



Compression Ratio

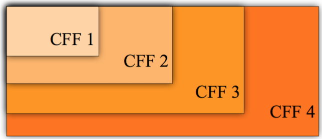
- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
  - number of signatures/size of the aggregate signature.



Compression Ratio

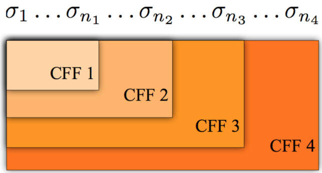
- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
  - number of signatures/size of the aggregate signature.
- The larger $\rho(n)$ the better.

$$\sigma_1 \ldots \sigma_{n_1} \ldots \sigma_{n_2} \ldots \sigma_{n_3} \ldots \sigma_{n_4}$$



Compression Ratio

- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
  - number of signatures/size of the aggregate signature.
- The larger $\rho(n)$ the better.
- $\rho(n)$ depends on d.



Compression Ratio

- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
- **Traditional aggregation**:
  $\rho(n) = n \implies t = 1, d = 0.$

item	1	2	3	4	5	6
agg 1	1	1	1	1	1	1

- **No aggregation**:
  $\rho = 1 \implies t = n, d = n$

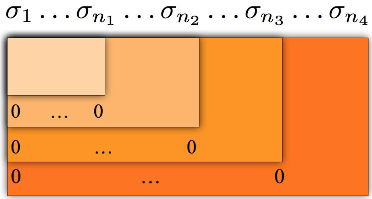
item	1	2	3	4	5	6
agg 1	1	0	0	0	0	0
agg 2	0	1	0	0	0	0
⋮			⋮			
agg 6	0	0	0	0	0	1

- **Fault-tolerant aggregation**: $\rho(n) \leq \frac{n}{\frac{d^2}{\log d} \log n}$.

Monotone family

- Solution with *Monotone families* [2]
- Avoid using unavailable signatures in new aggregates with 0 rows.

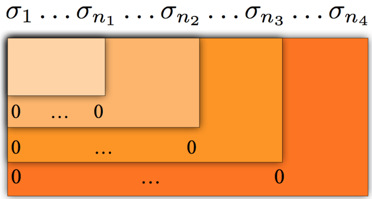
$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ 0 & W \end{pmatrix}$$



[2] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

- Solution with *Monotone families* [2]
- Avoid using unavailable signatures in new aggregates with 0 rows.

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ 0 & W \end{pmatrix}$$



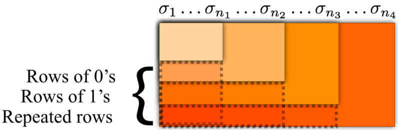
- Compression ratio: $\rho(n) = 1$ (number of rows is linear in n).
- Solved unbounded problem but impractical (constant ratio).

[2] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

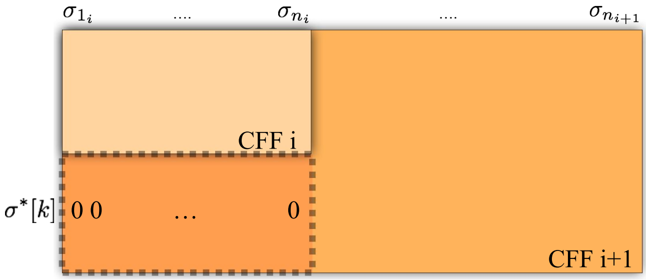
Our contribution:

- We define a more flexible family of matrices: *nested families.* [3]
- Z has rows of 0's, 1's, and repeated rows from $\mathcal{M}^{(l)}$.

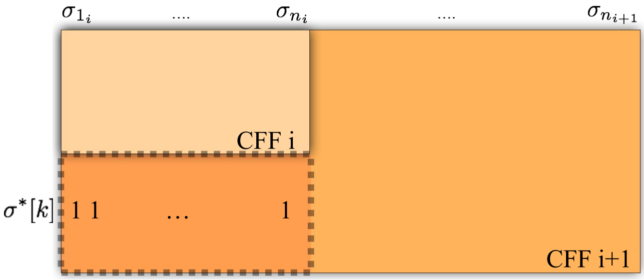
$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ Z & W \end{pmatrix}$$



[3] T. B. Idalino, L. Moura, TCS 2021.

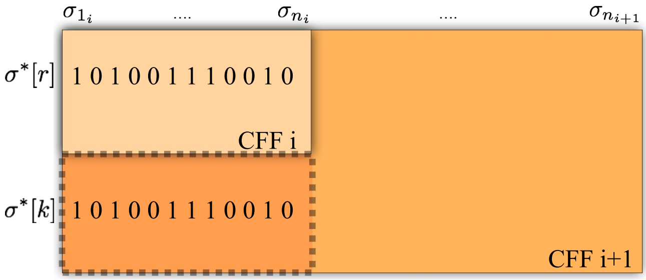


Row of 0's: $\sigma^*[k]$ is a regular aggregation.



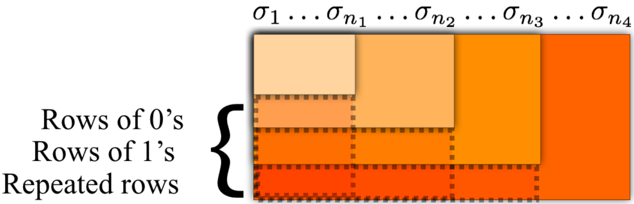
Row of 1's:

- Keep one extra aggregation $\sigma^*[0] = Agg(\sigma_i, \ldots, \sigma_{n_i})$;
- then $\sigma^*[k] = Agg(\sigma^*[0], \text{new signatures})$.



Repeated row r: $\sigma^*[k] = Agg(\sigma^*[r], \text{new signatures})$.

- We need constructions for nested families, with good increasing compression ratio
- Proposed 3 different constructions for $d = 1$ and general d



Case $d = 1$:

- Based on Sperner set systems.

1-CFF(6,20) Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$test_1$	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	0
$test_3$	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1
$test_4$	0	0	1	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0	1	1
$test_5$	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1	1	1
$test_6$	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Case $d = 1$:

- Based on Sperner set systems.

1-CFF(6,20) Matrix

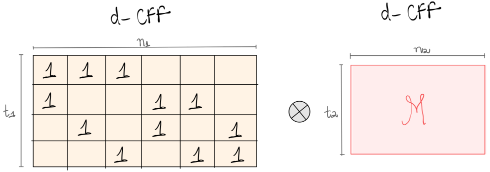
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$test_1$	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	0
$test_3$	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1
$test_4$	0	0	1	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0	1	1
$test_5$	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1	1	1
$test_6$	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

- We increase t as necessary and fill the matrix accordingly.
- $\rho(n) = \frac{n}{\log_2 n} \rightarrow$ meets the upper bound.

General d (Construction 1):

Kronecker Product

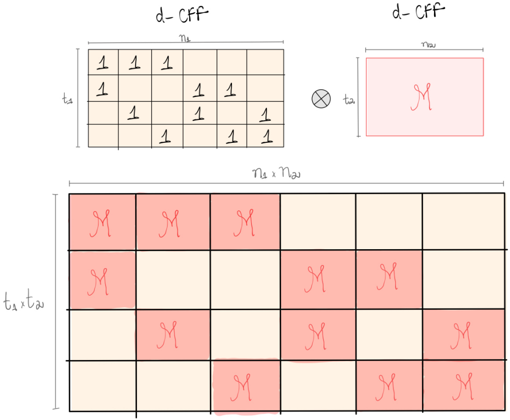
$d - CFF(t_1, n_1) \otimes d - CFF(t_2, n_2) = d - CFF(t_1 \times t_2, n_1 \times n_2)$



General d **(Construction 1)**:

KRONECKER PRODUCT

$$d - CFF(t_1, n_1) \otimes d - CFF(t_2, n_2) = d - CFF(t_1 \times t_2, n_1 \times n_2)$$



General d (Construction 1):

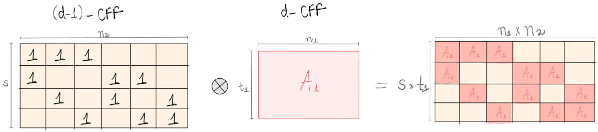
Iterating the Step

Iterating the step we get a nested family with

$$\rho(n) = \frac{n}{n^{1/c}} = n^{1-1/c}.$$

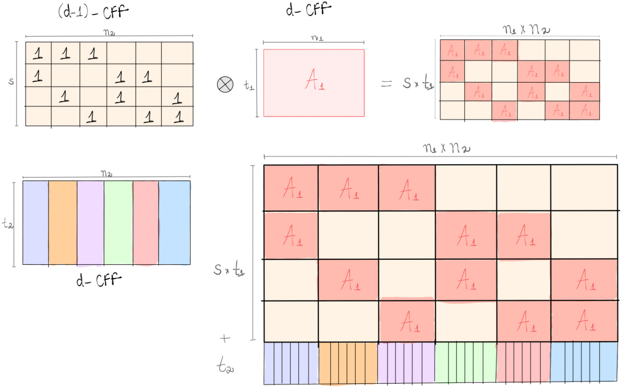
General d (Construction 2):

$$(d-1)\text{-CFF}(s, n_2) \otimes d\text{-CFF}(t_1, n_1)) \text{ plus } d\text{-CFF}(t_2, n_2)$$
$$= \quad d\text{-CFF}(s \times t_1 + t_2, n_2 \times n_1)$$



General d (Construction 2):

$$(d-1)\text{-CFF}(s, n_2) \otimes d\text{-CFF}(t_1, n_1)) \text{ plus } d\text{-CFF}(t_2, n_2)$$
$$= \quad d\text{-CFF}(s \times t_1 + t_2, n_2 \times n_1)$$



General d (Construction 2):

ITERATING THE STEP

Iterating the step (in a specific way) we get a nested family with

$$\rho(n) = \frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}.$$

With Nested families:

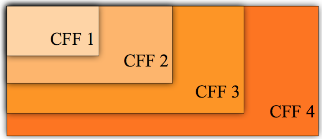
- Make fault-tolerant aggregation of signatures more practical.
  - Allow increase on the number n of signatures.
  - Reasonable aggregate signature size.

d	$\rho(n)$	**Construction**
0	n	Traditional
1	$\frac{n}{\log_2 n}$	Sperner
d	$\frac{n}{n^{1/c}}$	Construction 1
d	$\frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}$	Construction 2
d	1	Hartung et al. [4]

[4] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

What else?

$$\sigma_1 \dots \sigma_{n_1} \dots \sigma_{n_2} \dots \sigma_{n_3} \dots \sigma_{n_4}$$



- Increases in n may increase d too.
- Nested and monotone families do not allow increases on d.

- Generalization of monotone and nested: *embedding families*.[5]
- No requirements for Z.

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ Z & W \end{pmatrix}$$

- Application in broadcast encryption and authentication.
- Constructions based on polynomials over finite fields and extension fields.

[5] T. B. Idalino, L. Moura, to appear in Advances in Mathematics of Communications, nov 2019.

Construction (K&S 1964, E,F&F 1985)

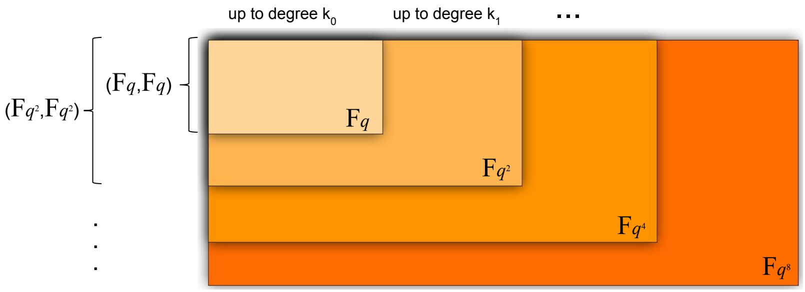
Let q be a prime power and k be a positive integer. If $q \geq dk + 1$ then there exists a d-CFF$((q^2, q^{k+1})$.

Note $t = q^2 = n^{\frac{2}{k+1}}$

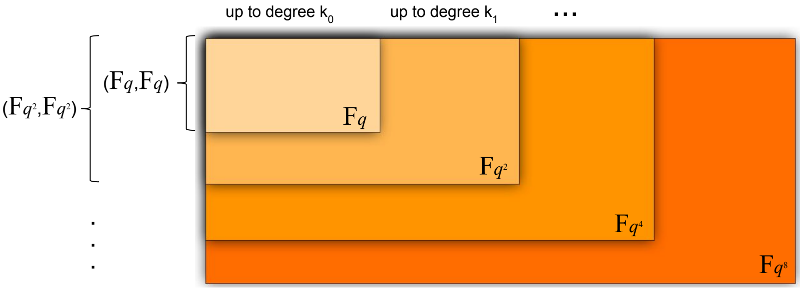
Example of for $q = 3$, $k = 1$: 1-CFF$(6, 9)$ and a 2-CFF$(9, 9)$:

	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
$(0,0)$	1	0	0	1	0	0	1	0	0
$(0,1)$	0	1	0	0	1	0	0	1	0
$(0,2)$	0	0	1	0	0	1	0	0	1
$(1,0)$	1	0	0	0	0	1	0	1	0
$(1,1)$	0	1	0	1	0	0	0	0	1
$(1,2)$	0	0	1	0	1	0	1	0	0
$(2,0)$	1	0	0	0	1	0	0	0	1
$(2,1)$	0	1	0	0	0	1	1	0	0
$(2,2)$	0	0	1	1	0	0	0	1	0

- Start with $\mathbb{F}_q$ and grow the construction with extension fields.
  - Tower of finite fields.
- Order rows and columns to have an embedding family.



- Play with k_i, d_i, q^{2^i}, for $q^{2^i} \geq d_i k_i + 1$:
  - Focus on d increases (fix k);
  - Focus on better compression ratio (fix d);
  - Build monotone families with increasing $\rho(n)$ (fix d and k).



Prioritizing d increase

- Fix k and increase d_i to its maximum.
- $q^{2^i} \geq d_i k + 1$
- $\rho(n) = n^{1-\frac{2}{k+1}}$
- $d \sim \frac{n^{1/k+1}}{k}$

i	q	k	d	n	t	n/t
0	4	2	1	64	12	5.33
1	16	2	7	4096	240	17.06
2	256	2	127	16777216	65280	257.00
3	65536	2	32767	281474976710656	4294901760	65537.00

Prioritize ratio increases

Prioritizing ratio increase

- Fix d and increase k_i to its maximum.
- $q^{2^i} \geq dk_i + 1$
- $\rho(n) = \frac{n}{\log n}$
  - Because $n = q^{k+1}$, $t = (dk + 1)q$.

i	q	k	d	n	t	n/t
0	4	1	2	16	12	1.33
1	16	7	2	4294967296	240	17895697.07
2	256	127	2	256^{128}	65280	2.75×10^{303}
3	65536	32767	2	65536^{32768}	4294901760	6.04×10^{157816}

MONOTONE FAMILIES

- Fix d and k.
- Select specific blocks of rows.
- We get monotone families with $\frac{n}{t} = \frac{n}{qn^{1/k+1}}$, which is $O(n^{1-\frac{1}{k+1}})$.

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ 0 & W \end{pmatrix}$$

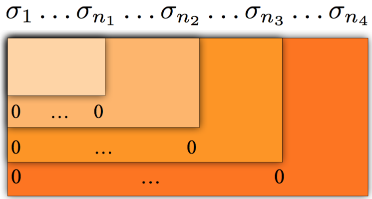


TABLE: Embedding families: Summary of results for $k \geq 2$.

k	d	$\rho(n)$	Feature
fixed	$d \sim \frac{n^{1/(k+1)}}{k}$	$n^{1-\frac{2}{k+1}}$	increasing d
increasing	fixed	$\frac{n}{\log n}$	optimal ratio
fixed	fixed	$n^{1-\frac{1}{k+1}}$	monotone

Conclusion

Different applications require different properties of CFFs.

- Explore dynamic applications with increasing n and d.
- Good compression ratios.

	d	n
d-CFFs	fixed	fixed
Monotone	fixed	increasing
Nested	**fixed**	**increasing**
Embedding	**increasing**	**increasing**

Future Work

- Constructions with better compression ratio.
- Compression ratio bounds on monotone and nested families ($d \geq 2$).
  - $\rho(n) \leq \frac{n}{\frac{d^2}{\log d} \log n}$
- New constructions of embedding families with smoother compression ratio.
  - Gradual increases of n.
- Other aspects of CFFs to be explored.
  - Mixed properties and applications.

Thank You!

thais.bardini@ufsc.br

References I

[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, *Aggregate and verifiably encrypted signatures from bilinear maps*, in EUROCRYPT 2003.

[2] D. Du, F. Hwang. *Combinatorial group testing and its applications*. In World Scientific, 2000.

[3] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp. *Fault-Tolerant Aggregate Signatures.* In Public-Key Cryptography – PKC 2016, pages 331–356, 2016.

[4] T.B. Idalino. *Using combinatorial group testing to solve integrity issues*. Master's thesis, 2015.

[5] T.B. Idalino and L. Moura, *Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families*, in Lecture Notes in Computer Science, IWOCA 2018.

[6] T.B. Idalino and L. Moura, *Embedding sequences of cover-free families and cryptographical applications*. Advances in Mathematics of Communications, 2019.

[7] P. C. Li, G. H. J. van Rees, R. Wei. *Constructions of 2-cover-free families and related separating hash families*. in Journal of Combinatorial Designs, 2006.

[8] E. Sperner. *Ein Satz über Untermengen einer endlichen Menge.* in Mathematische Zeitschrift, 1928.