Fault tolerance in cryptography using cover-free families

Thais Bardini Idalino
Universidade Federal de Santa Catarina, Brazil

Lucia Moura
University of Ottawa, Canada

Congreso Latinoamericano de Matemáticos (CLAM) 2021

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

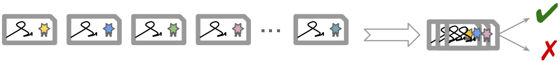
Outline

- 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.

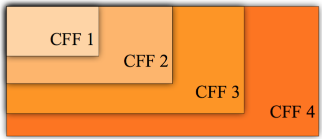


- Cover-free families to provide fault-tolerance.

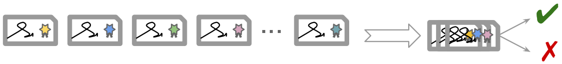
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg1:	1	1	1	0	0	0
agg2:	1	0	0	1	1	0
agg3:	0	1	0	1	0	1
agg4:	0	0	1	0	1	1

- Explore different aspects of cover-free families.

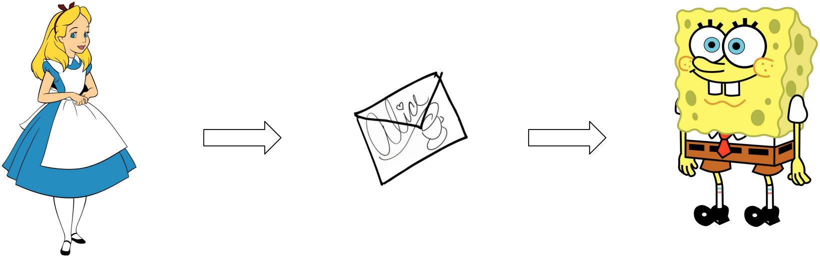
$\sigma_1 \ldots \sigma_{n_1} \ldots \sigma_{n_2} \ldots \sigma_{n_3} \ldots \sigma_{n_4}$



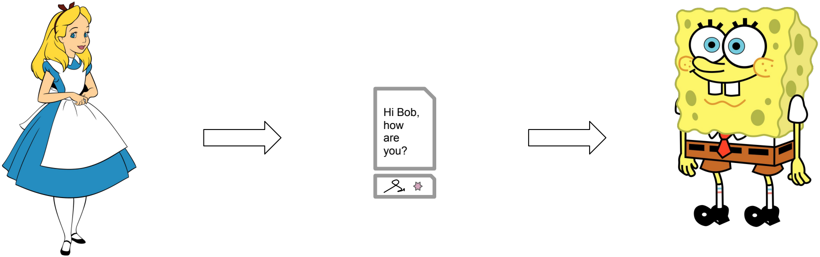
The problem



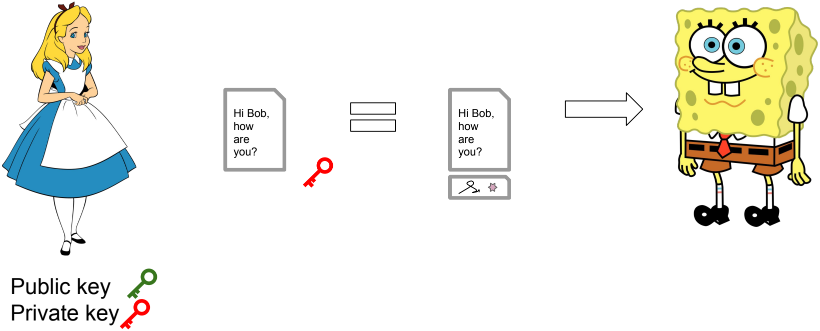
Digital signatures



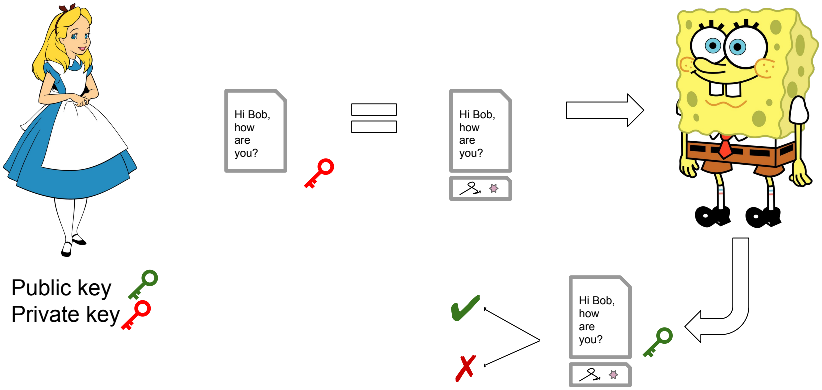
Digital signatures



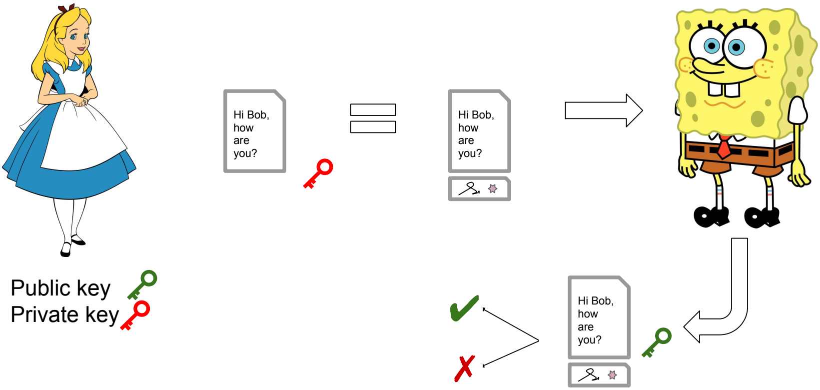
Digital signatures



Digital signatures

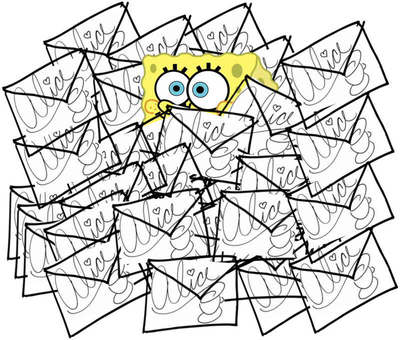


Digital signatures

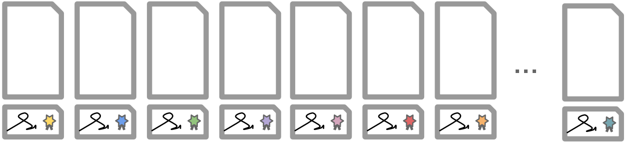


- Allows Bob to verify that the message was not modified during transmission (**integrity**), and that Alice in fact signed it (**authenticity**).

What happens when we have thousands of messages and signatures?

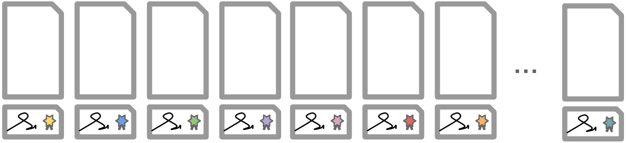


- What happens when we have thousands of msgs/signatures?

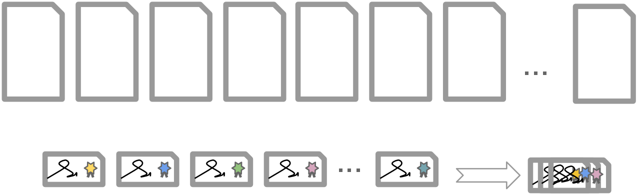


[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Eurocrypt 2003.

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

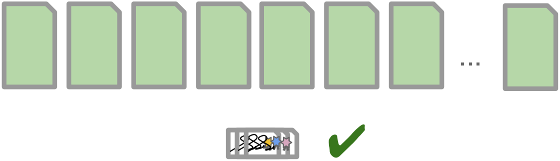


- *Aggregation of signatures*, Boneh et al. (2003)[1].



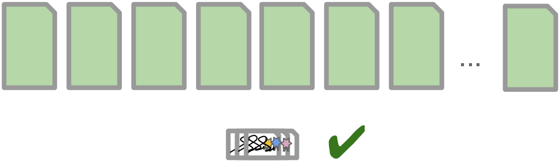
[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Eurocrypt 2003.

- Saves on storage, communication and verification time.

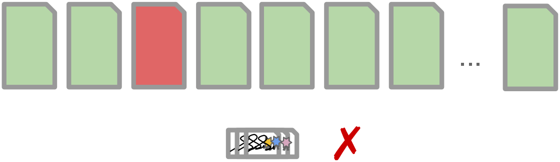


Aggregation of signatures

- Saves on storage, communication and verification time.

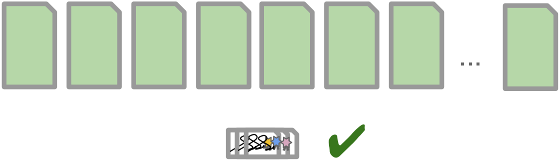


- One invalid signature invalidates the entire aggregate.

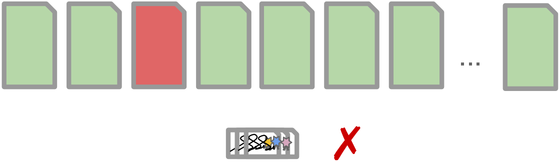


Aggregation of signatures

- Saves on storage, communication and verification time.



- One invalid signature invalidates the entire aggregate.



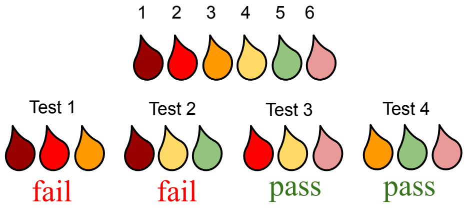
- Use d-cover-free families to provide fault-tolerance.

Cover-free families

	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

Combinatorial group testing

Problem: Identify d defective elements from a set of n elements pooled into t groups.
Solution: Test the t groups, instead of individual elements.
Objective: Give n, d, minimize t.

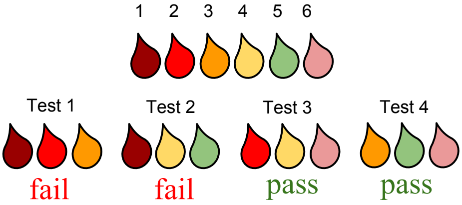


Cover-free families

Combinatorial group testing

Problem: Identify d defective elements from a set of n elements pooled into t groups.
Solution: Test the t groups, instead of individual elements.
Objective: Give n, d, minimize t.

- Use a d-CFF(t, n) with t rows and n columns.



1-CFF(4,6) Matrix

$$\begin{array}{c} \\ \text{test}_1 \\ \text{test}_2 \\ \text{test}_3 \\ \text{test}_4 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Example $d = 1$ defectives

item	1	2	3	4	5	6	
input:	?	?	?	?	?	?	result:
test1:	1	1	1	0	0	0	?
test2:	1	0	0	1	1	0	?
test3:	0	1	0	1	0	1	?
test4:	0	0	1	0	1	1	?

Defective: item ?

Example $d = 1$ defectives

item	1	2	3	4	5	6	
input:	?	?	?	?	?	?	result:
test1:	1	1	1	0	0	0	1
test2:	1	0	0	1	1	0	0
test3:	0	1	0	1	0	1	0
test4:	0	0	1	0	1	1	1

Defective: item ?

Example $d = 1$ defectives

item	1	2	3	4	5	6	
input:	0	0	?	0	0	0	result:
test1:	1	1	1	0	0	0	1
test2:	1	0	0	1	1	0	0
test3:	0	1	0	1	0	1	0
test4:	0	0	1	0	1	1	1

Defective: item ?

Example $d = 1$ defectives

item	1	2	3	4	5	6	
input:	0	0	1	0	0	0	result:
test1:	1	1	1	0	0	0	1
test2:	1	0	0	1	1	0	0
test3:	0	1	0	1	0	1	0
test4:	0	0	1	0	1	1	1

Defective: item 3

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	?	?	?	?	?	?	?	?	?	?	?	?	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	?
t2	1	0	0	0	1	0	0	1	0	0	1	0	?
t3	1	0	0	0	0	1	0	0	1	0	0	1	?
t4	0	1	0	1	0	0	0	0	1	0	1	0	?
t5	0	1	0	0	1	0	1	0	0	0	0	1	?
t6	0	1	0	0	0	1	0	1	0	1	0	0	?
t7	0	0	1	1	0	0	0	1	0	0	0	1	?
t8	0	0	1	0	1	0	0	0	1	1	0	0	?
t9	0	0	1	0	0	1	1	0	0	0	1	0	?

Defective: items ?

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	?	?	?	?	?	?	?	?	?	?	?	?	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items ?

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	?	0	0	0	0	0	0	0	0	?	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items ?

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

Example $d = 2$ defectives

item	1	2	3	4	5	6	7	8	9	10	11	12	
input	0	0	1	0	0	0	0	0	0	0	0	1	result:
t1	1	0	0	1	0	0	1	0	0	1	0	0	0
t2	1	0	0	0	1	0	0	1	0	0	1	0	0
t3	1	0	0	0	0	1	0	0	1	0	0	1	1
t4	0	1	0	1	0	0	0	0	1	0	1	0	0
t5	0	1	0	0	1	0	1	0	0	0	0	1	1
t6	0	1	0	0	0	1	0	1	0	1	0	0	0
t7	0	0	1	1	0	0	0	1	0	0	0	1	1
t8	0	0	1	0	1	0	0	0	1	1	0	0	1
t9	0	0	1	0	0	1	1	0	0	0	1	0	1

Defective: items 3 and 12

A d-CFF(t, n) is a $t \times n$ binary matrix where every set of $d+1$ columns contains a permutation submatrix of order $d+1$.

Using a d-cover-free family, no matter where the defectives are...

$$
\begin{array}{cccccccccccc}
 & & c & & & c_1 & , & \ldots & , & c_d & & \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\end{array}
$$

we can identify all good items.

(as long as we have at most d defectives)

When $d = 1$:

- no column is covered by any other
- no subset contains any other
- Sperner set systems

item	1	2	3	4	5	6
test1:	1	1	1	0	0	0
test2:	1	0	0	1	1	0
test3:	0	1	0	1	0	1
test4:	0	0	1	0	1	1

When $d = 1$:

- Given n, choose $t = \min\{s : \binom{s}{\lfloor s/2 \rfloor} \geq n\}$.
- List the collection of all the $\lfloor t/2 \rfloor$-subsets of $\{1, \ldots, t\}$.

Example $n = 6, t = 4, d = 1$
2-**subsets of** $\{1, 2, 3, 4\}$: $\{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}\}$

item	1	2	3	4	5	6
test1:	1	1	1	0	0	0
test2:	1	0	0	1	1	0
test3:	0	1	0	1	0	1
test4:	0	0	1	0	1	1

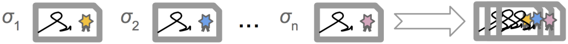
- Minimize t for given n and d.
  - Lower bound for $d \geq 2$: $t \geq c \frac{d^2}{\log d} \log n$. [2]

- For $d = 1$:
  - Sperner system.
  - $t \sim \log n$.

- For general d:
  - Direct construction from finite fields, codes, SHF, OAs, etc.
  - Probabilistic construction with $t = \Theta((d+1)^2 \ln n)$.

[2] Z. Füredi. On r-Cover-free Families. Journal of Combinatorial Theory, 1996.

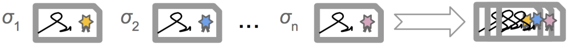
- Traditional aggregation of signatures



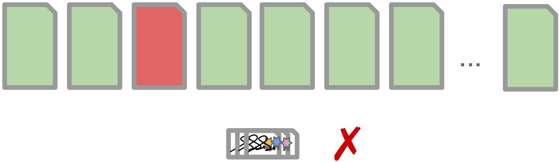
[3] T. B. Idalino. Using combinatorial group testing to solve integrity issues. Master's thesis, 2015.

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

- Traditional aggregation of signatures



- One invalid signature invalidates the entire aggregate.

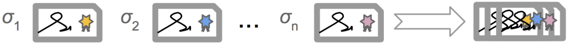


[3] T. B. Idalino. Using combinatorial group testing to solve integrity issues. Master's thesis, 2015.

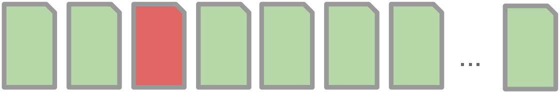
[4] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

Recall our problem

- Traditional aggregation of signatures



- One invalid signature invalidates the entire aggregate.



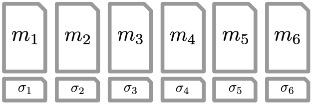
- Use d-CFFs to provide fault-tolerance. [3,4]

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

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

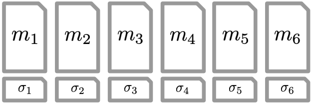
Fault tolerance with d-CFFs

- 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.



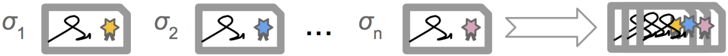
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	
agg 1:	1	1	1	0	0	0	$\sigma^*[1] = \textbf{Agg}(\sigma_1, \sigma_2, \sigma_3)$
agg 2:	1	0	0	1	1	0	$\sigma^*[2] = \textbf{Agg}(\sigma_1, \sigma_4, \sigma_5)$
agg 3:	0	1	0	1	0	1	$\sigma^*[3] = \textbf{Agg}(\sigma_2, \sigma_4, \sigma_6)$
agg 4:	0	0	1	0	1	1	$\sigma^*[4] = \textbf{Agg}(\sigma_3,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

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

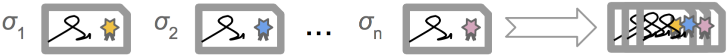
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	result:
agg 1:	1	1	1	0	0	0	✗
agg 2:	1	0	0	1	1	0	✓
agg 3:	0	1	0	1	0	1	✓
agg 4:	0	0	1	0	1	1	✗

Invalid signature: σ_3

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



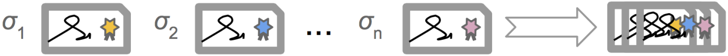
- **Before**: dynamically aggregate signatures as they arrive.



- **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.

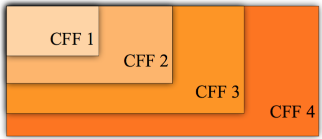
	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
agg 1:	1	1	1	0	0	0
agg 2:	1	0	0	1	1	0
agg 3:	0	1	0	1	0	1
agg 4:	0	0	1	0	1	1

- Impractical for applications where signatures are dynamically arriving.

Question

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.

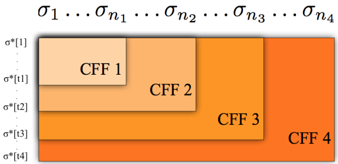
Unbounded aggregation of signatures

- **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

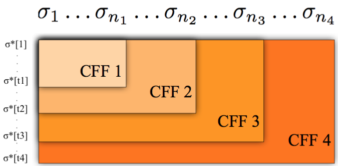
$$
\begin{array}{c c}
 & \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{array}{c} \text{test}_1 \\ \text{test}_2 \\ \text{test}_3 \\ \text{test}_4 \\ \text{test}_5 \end{array} &
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\end{array}
$$



- **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

$$
\begin{array}{c}
\\ \text{test}_1 \\ \text{test}_2 \\ \text{test}_3 \\ \text{test}_4 \\ \text{test}_5
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\end{array}
$$

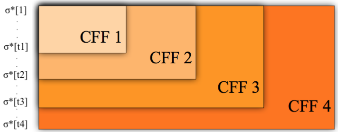


- **Problem:** Fault-tolerant aggregation of signatures with unknown n.
- **Solution:** Increase the d-CFF to hold extra signatures.
- Create a special sequence of d-CFF matrices.
  - Large matrices contain small matrices.
  - Avoid using unavailable signatures in the new aggregates.

1-CFF(5,10) Matrix

$$
\begin{array}{c c}
 & \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\begin{array}{c} \text{test}_1 \\ \text{test}_2 \\ \text{test}_3 \\ \text{test}_4 \\ \text{test}_5 \end{array} &
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\end{array}
$$

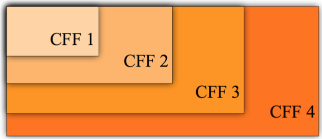
$\sigma_1 \ldots \sigma_{n_1} \ldots \sigma_{n_2} \ldots \sigma_{n_3} \ldots \sigma_{n_4}$



Compression Ratio

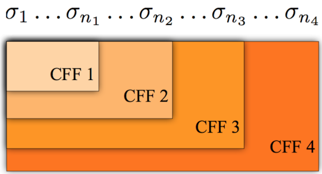
- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$

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



Compression Ratio

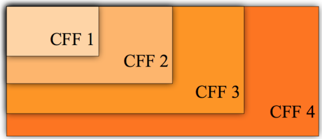
- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
  - number of signatures/size of the aggregate signature.



Compression Ratio

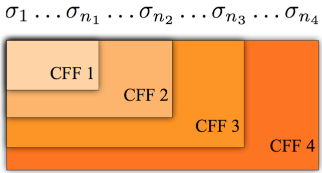
- **Compression ratio**: $\rho(n)$ iff $\frac{n}{t}$ is $\Theta(\rho(n))$
  - number of signatures/size of the aggregate signature.
- The larger $\rho(n)$ the better.

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



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**:
  $\rho = 1 \implies t = n, d = n$

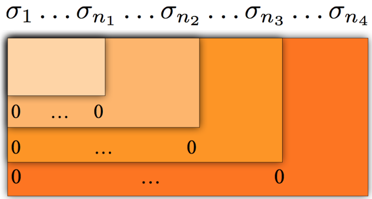
item	1	2	3	4	5	6
agg 1	1	0	0	0	0	0
agg 2	0	1	0	0	0	0
$\vdots$			$\vdots$			
agg 6	0	0	0	0	0	1

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

Monotone family

- Solution with *Monotone families* [5]
- Avoid using unavailable signatures in new aggregates with 0 rows.

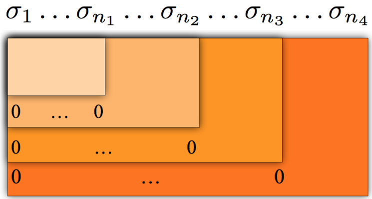
$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ 0 & W \end{pmatrix}$$



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

- Solution with *Monotone families* [5]
- 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).

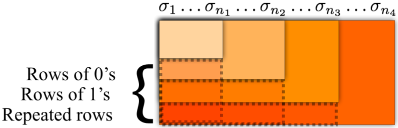
[5] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

Nested family
Definition

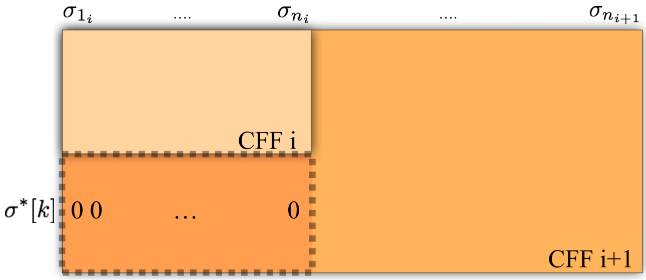
Our contribution:

- We define a more flexible family of matrices: *nested families*. [6]
- Z has rows of 0's, 1's, and repeated rows from $\mathcal{M}^{(l)}$.

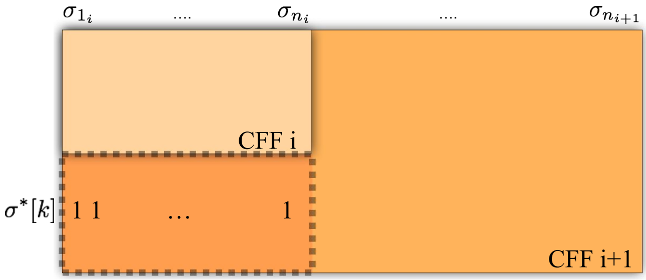
$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ Z & W \end{pmatrix}$$



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

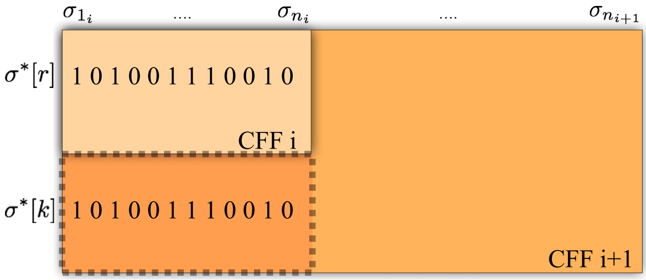


Row of 0's: $\sigma^*[k]$ is a regular aggregation.



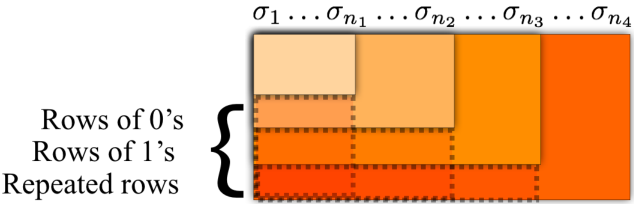
Row of 1's:

- 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

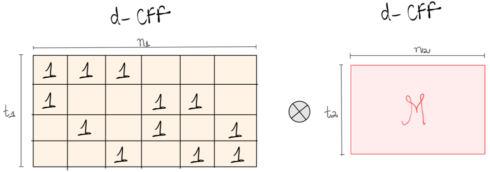
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$test_1$	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0
$test_2$	1	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	0
$test_3$	0	1	0	1	0	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1
$test_4$	0	0	1	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0	1	1
$test_5$	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	1	1	1
$test_6$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

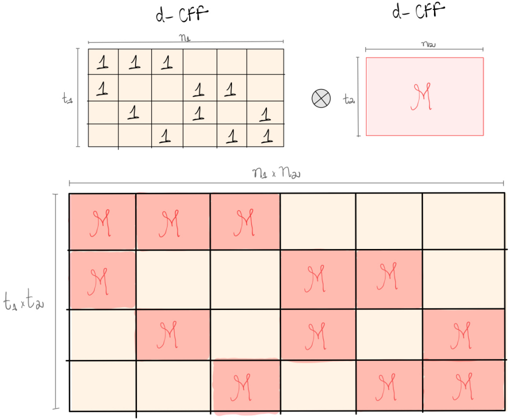
$$d\text{-}CFF(t_1, n_1) \otimes d\text{-}CFF(t_2, n_2) = d\text{-}CFF(t_1 \times t_2, n_1 \times n_2)$$



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):

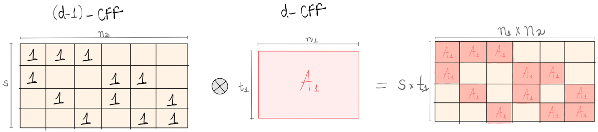
Iterating the Step

Iterating the step we get a nested family with

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

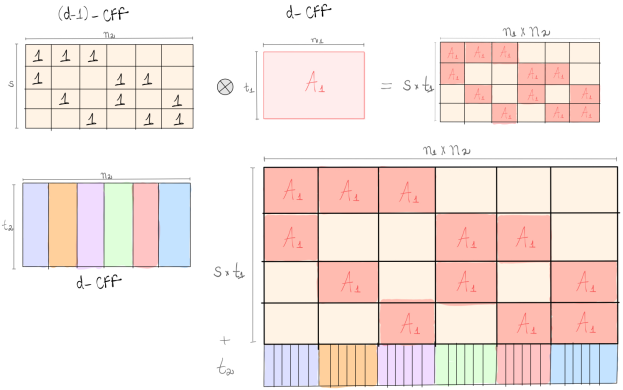
General d (Construction 2):

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



General d (Construction 2):

$(d-1)$-CFF$(s, n_2) \otimes d$-CFF$(t_1, n_1))$ plus d-CFF(t_2, n_2)
$= \quad d$-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

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

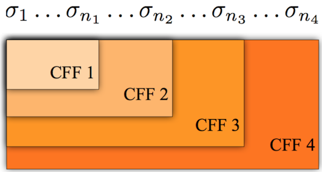
With Nested families:

- 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. [7]

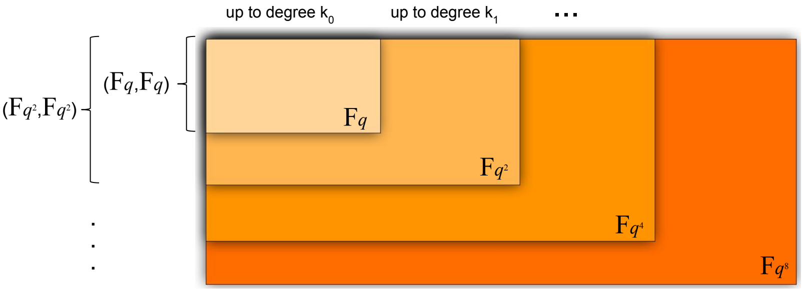
[7] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp, PKC 2016.

What else?



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

- Generalization of monotone and nested: *embedding families*.[8]
- Constructions based on polynomials over finite fields and extension fields.



[8] T. B. Idalino, L. Moura, Mathematics of Communications, 2019.

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 ($d \geq 2$).
  - $\rho(n) \leq \frac{n}{\frac{d^2}{\log d} \log n}$
- Other aspects of CFFs to be explored.
  - Mixed properties and applications.

thais.bardini@ufsc.br

[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.