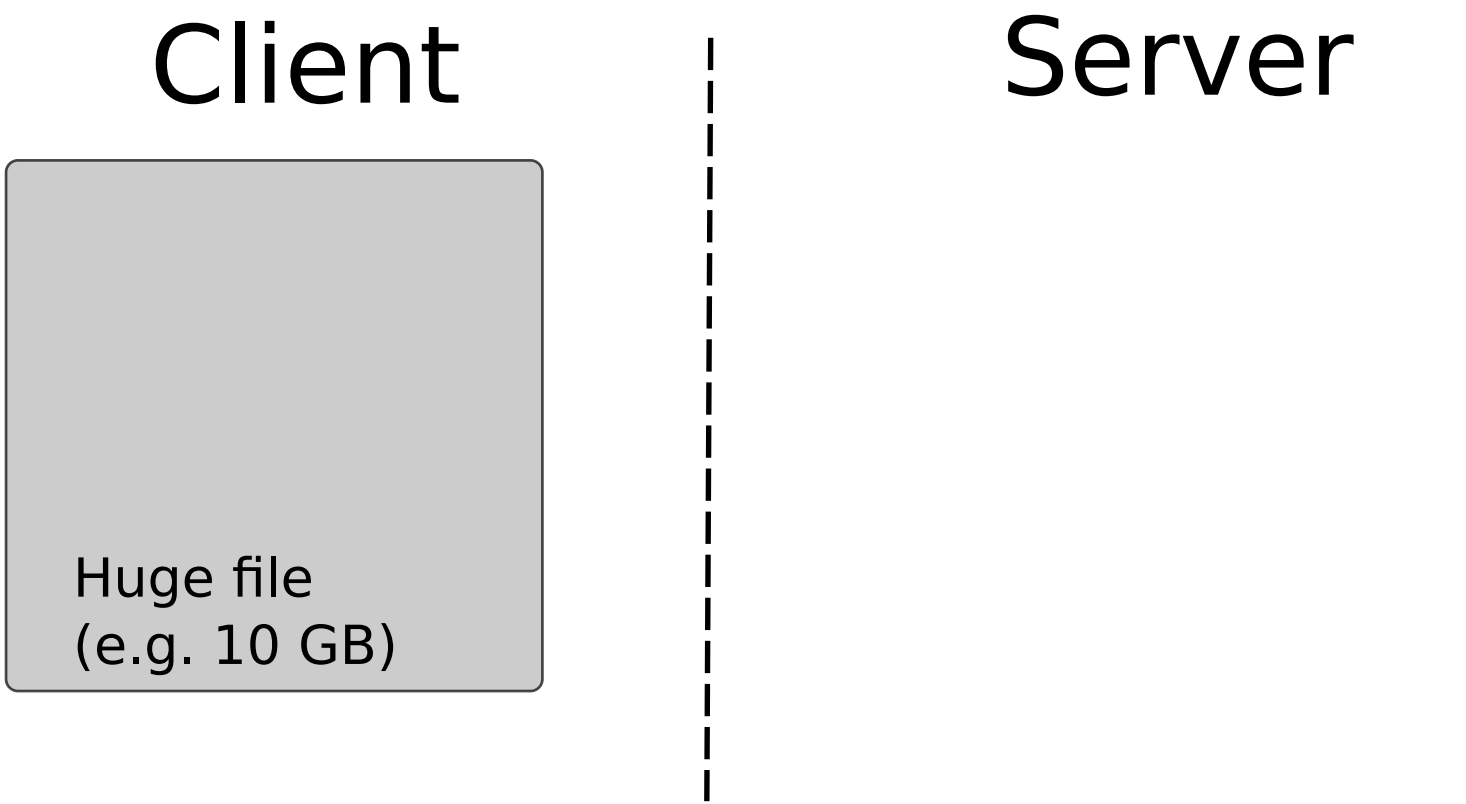
Proofs of Retrievability Using Locally Decodable Codes

Julien Lavauzelle, Françoise Levy-dit-Vehel

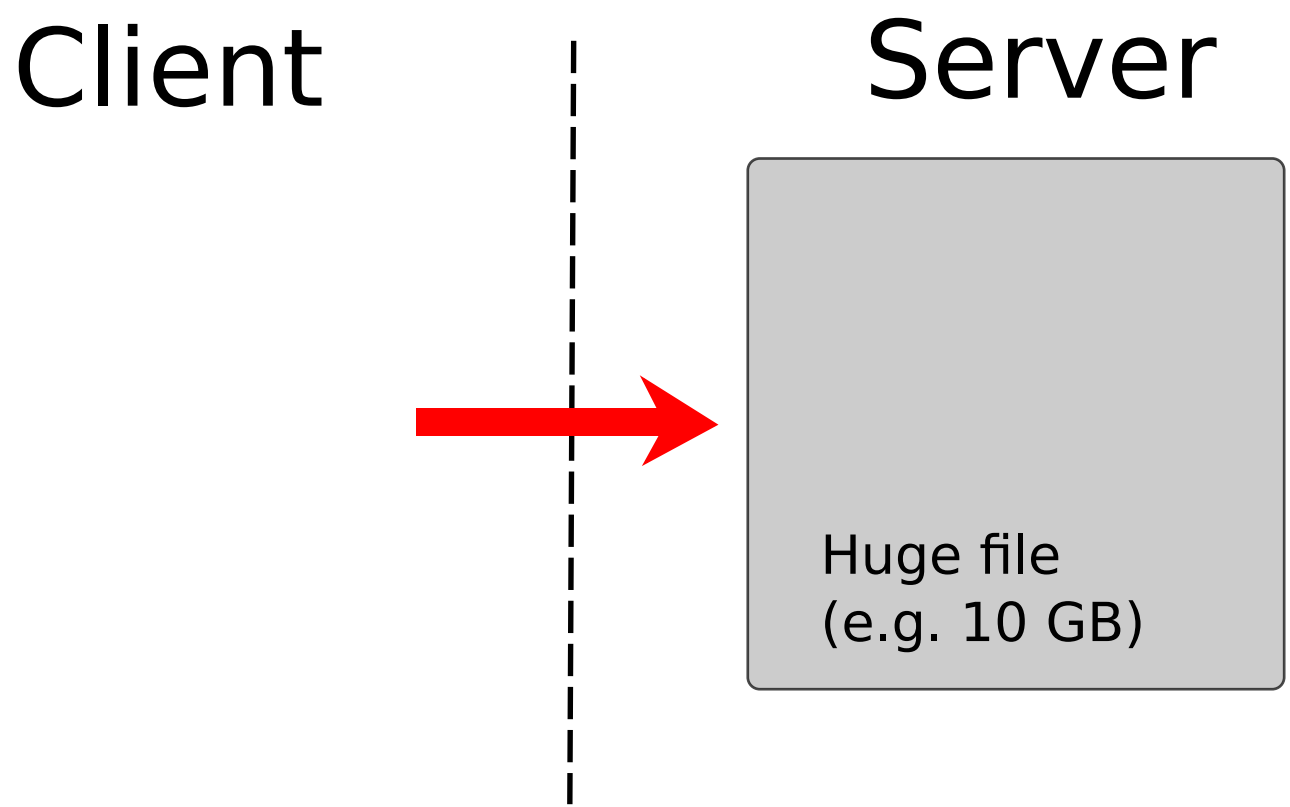
GRACE team, LIX & INRIA, Palaiseau, France

2016 IEEE International Symposium on Information Theory
July 13, 2016

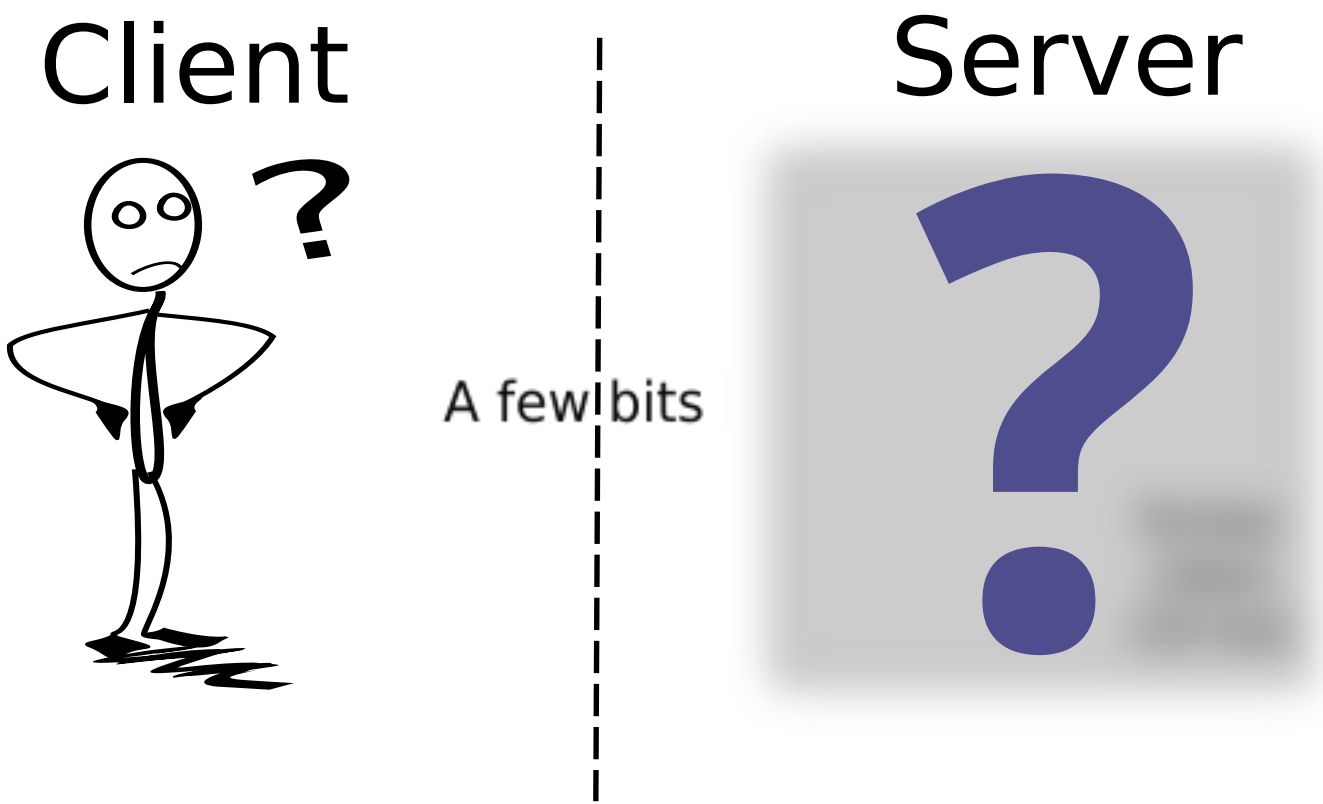
Issue



Issue



Issue



Is the file retrievable?

Definition (Proof of Retrievability)

Proof of Retrievability (PoR) = 3 procedures:

- **Initialization** [*Client*] :

$$F \mapsto \textbf{Init}(F) = (\tilde{F}, \mathtt{data}_F)$$

Then, the client uploads $\tilde{F}$ on the server.

Definition (Proof of Retrievability)

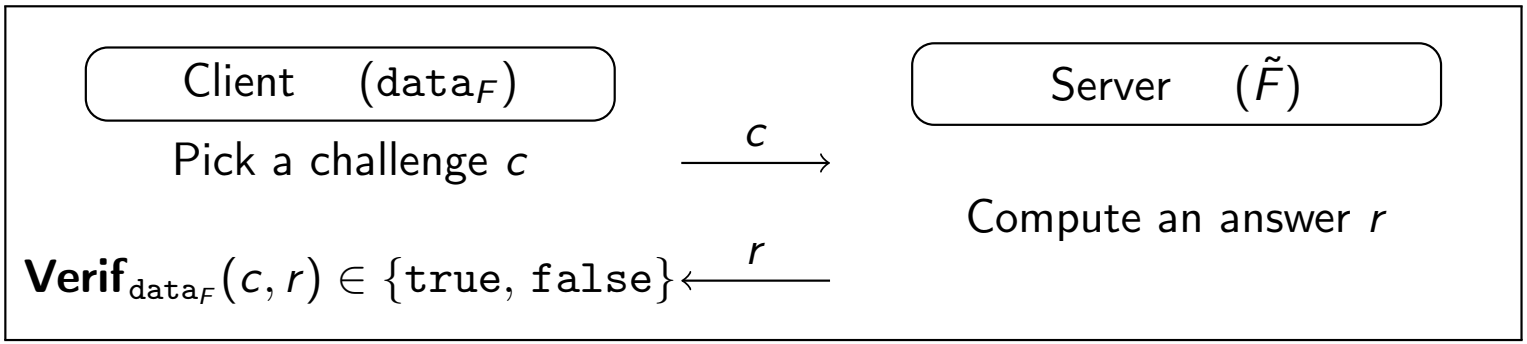
Proof of Retrievability (PoR) = 3 procedures:

- **Initialization** [*Client*] :

$$F \mapsto \textbf{Init}(F) = (\tilde{F}, \mathtt{data}_F)$$

Then, the client uploads $\tilde{F}$ on the server.

- **Verification** [*Client* $\longleftrightarrow$ *Server*] :



Definition (Proof of Retrievability)

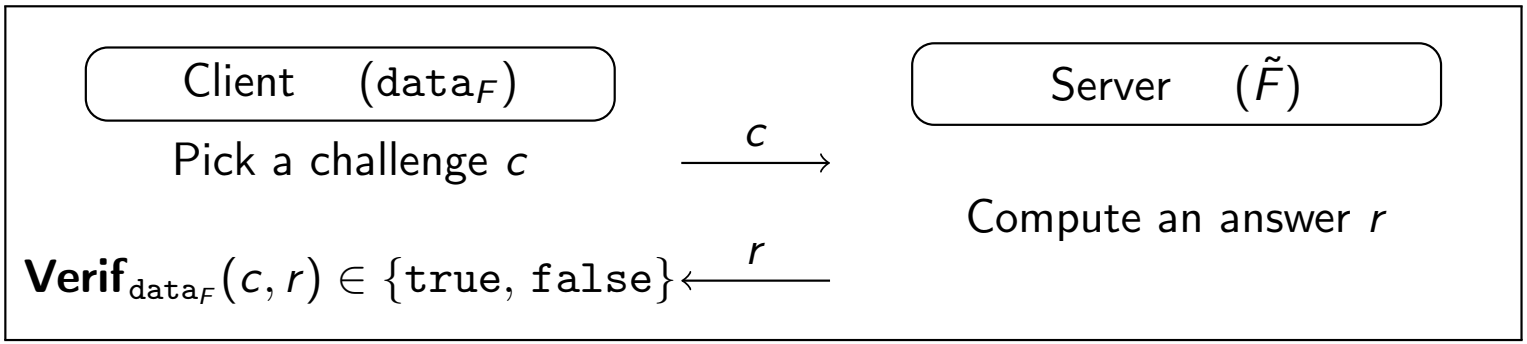
Proof of Retrievability (PoR) = 3 procedures:

- **Initialization** [*Client*] :

$$F \mapsto \textbf{Init}(F) = (\tilde{F}, \mathtt{data}_F)$$

Then, the client uploads $\tilde{F}$ on the server.

- **Verification** [*Client* $\longleftrightarrow$ *Server*] :



- **Extraction** [*Client* $\longleftrightarrow$ *Server*]. We want that

$$\textbf{Extract}(\mathtt{data}_F) = F \quad \text{holds w.h.p.}$$

A security model

Definition (τ-faulty server). Let $\tau \in [0,1]$, $\mathcal{P}$ be a PoR and X the distribution of challenges. A *τ-faulty server* $\mathcal{A}$ for $\mathcal{P}$ is an algorithm such that, for all encoded files $\tilde{F}$:

$$\mathbb{P}_{x \sim X}\left[\mathbf{Verif}_{\mathtt{data}_F}(x, \mathcal{A}(\tilde{F}, x)) = \mathtt{false}\right] < \tau\,.$$

Rem: this also includes malicious servers.

A security model

Definition (τ-faulty server). Let $\tau \in [0,1]$, $\mathcal{P}$ be a PoR and X the distribution of challenges. A *τ-faulty server* $\mathcal{A}$ for $\mathcal{P}$ is an algorithm such that, for all encoded files $\tilde{F}$:

$$\mathbb{P}_{x\sim X}\left[\textbf{Verif}_{\texttt{data}_F}(x, \mathcal{A}(\tilde{F}, x)) = \texttt{false}\right] < \tau\,.$$

Rem: this also includes malicious servers.

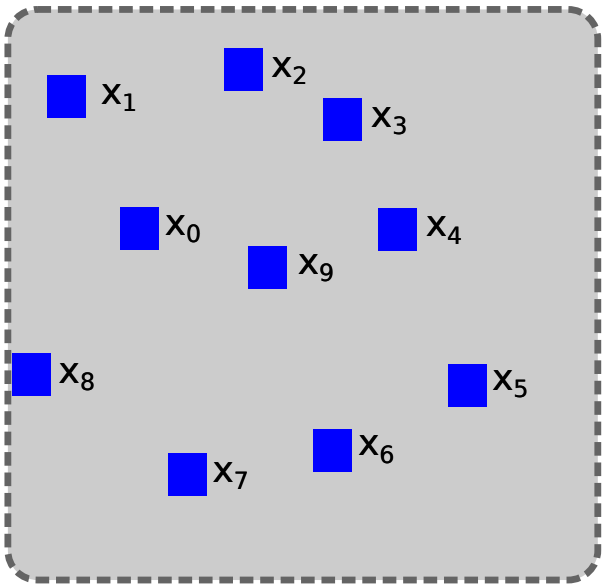
Definition (PoR soundness). Let $\tau, \varepsilon \in [0,1]$. A PoR is said (τ, ε)-sound if, for all τ-faulty servers $\mathcal{A}$ and all files F:

$$\mathbb{P}\left[\textbf{Extract}(\texttt{data}_F) = F\right] \geq 1 - \varepsilon\,,$$

where the probability is taken over extraction procedure randomness.

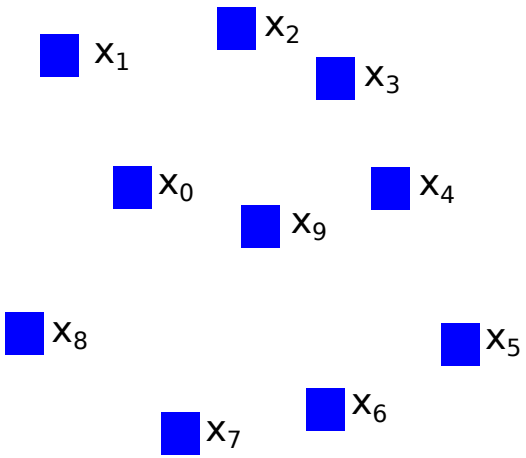
The seminal example

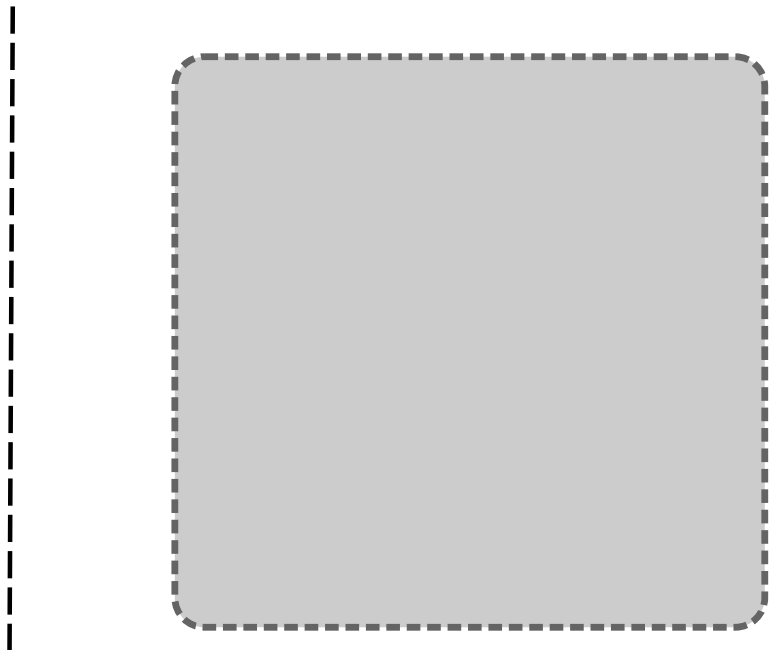
A. Juels, B. Kaliski Jr., *PORs: Proofs of Retrievability for large files.* in Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, USA, 2007.



The seminal example

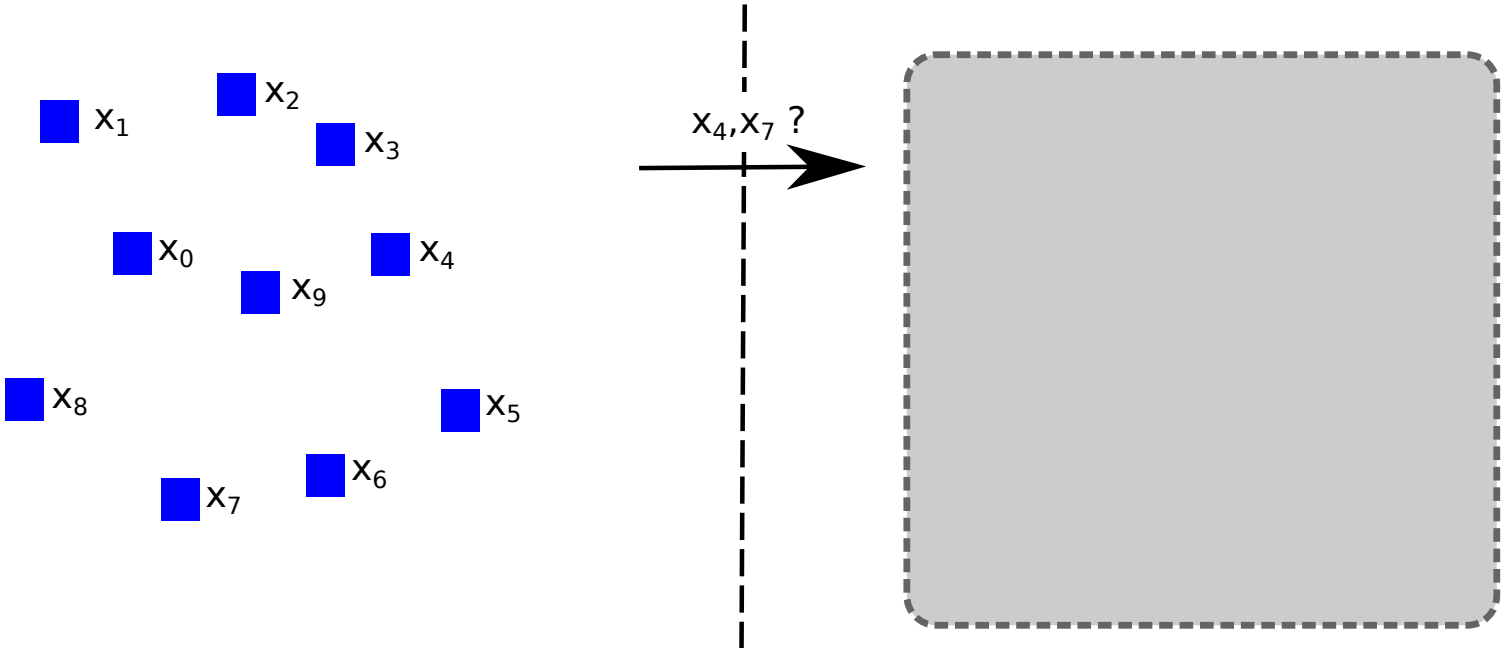
A. Juels, B. Kaliski Jr., *PORs: Proofs of Retrievability for large files.* in Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, USA, 2007.





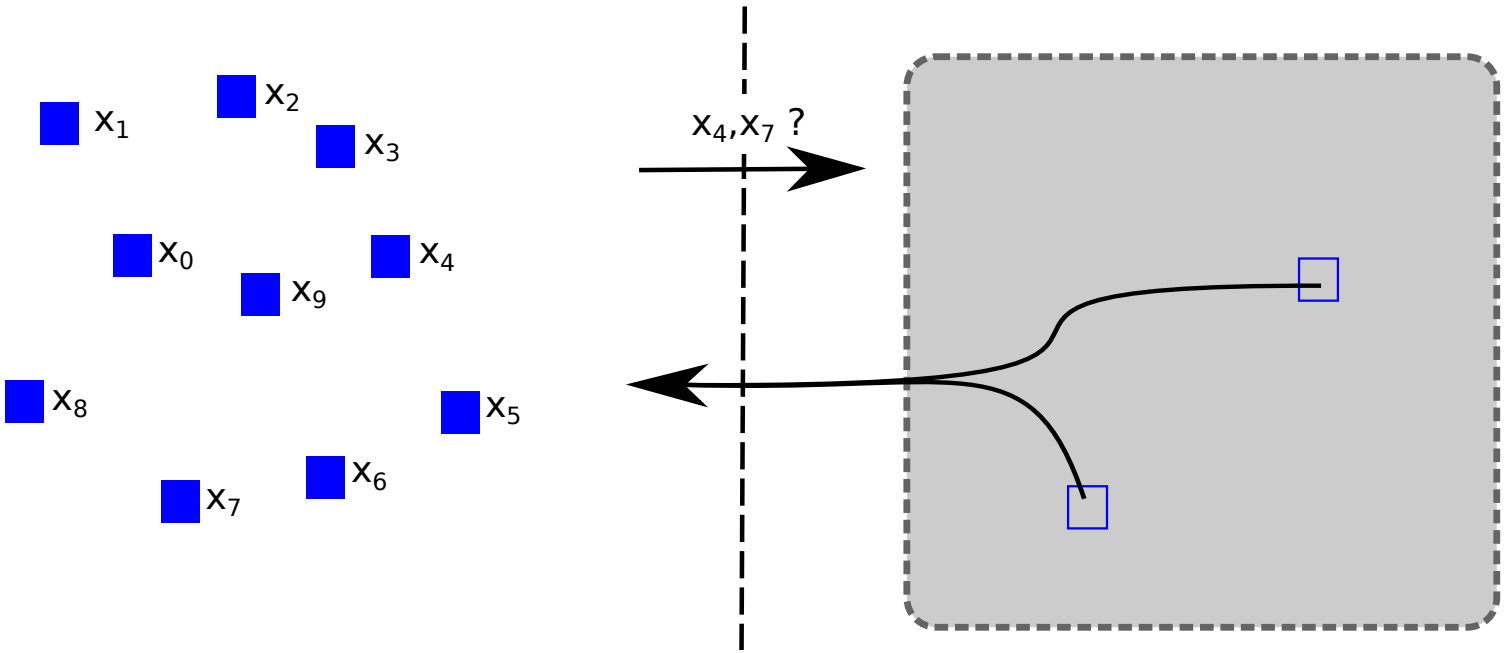
The seminal example

A. Juels, B. Kaliski Jr., *PORs: Proofs of Retrievability for large files.* in Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, USA, 2007.



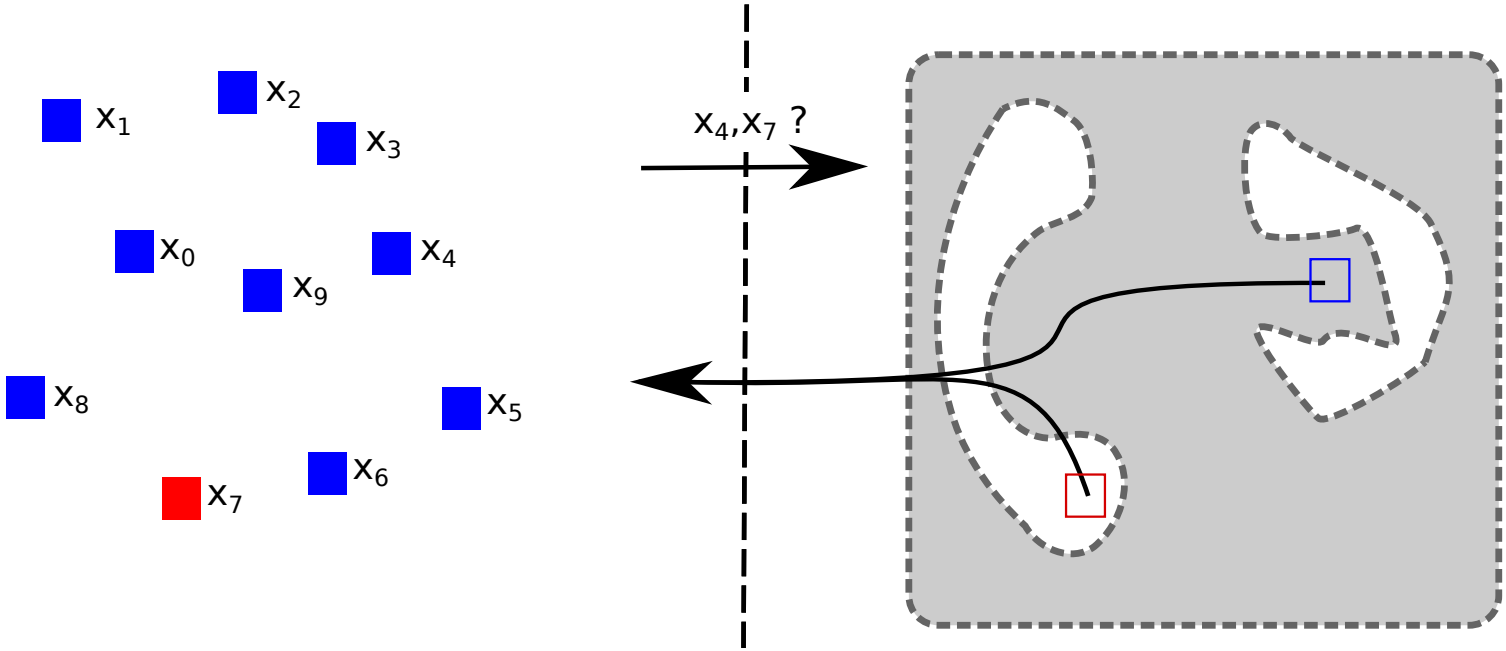
The seminal example

A. Juels, B. Kaliski Jr., *PORs: Proofs of Retrievability for large files.* in Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, USA, 2007.



The seminal example

A. Juels, B. Kaliski Jr., *PORs: Proofs of Retrievability for large files.* in Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, USA, 2007.



Other features

- Low communication;
- low server overhead and low client storage;
- low algorithmic complexity;
- unbounded use.

Towards structure verification

JK'07 main drawbacks:

- bounded use;
- quite big client storage.

Towards structure verification

JK'07 main drawbacks:

- bounded use;
- quite big client storage.

Our idea:

- check the **structure** of the file instead of file values;
- use **locally decodable codes** which provide a local structure.

Error-correcting codes

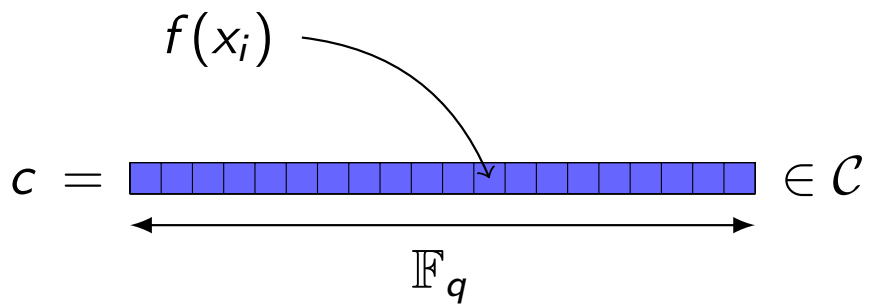
$\mathbb{F}_q = \{t_1, \ldots, t_q\}$ a finite field.

$$\begin{array}{rccc} \mathrm{ev}_1 : & \mathbb{F}_q[T] & \to & \mathbb{F}_q^q \\ & f & \mapsto & (f(t_1), \ldots, f(t_q)) \end{array}$$

$$\begin{array}{rccc} \mathrm{ev}_m : & \mathbb{F}_q[X_1, \ldots, X_m] & \to & \mathbb{F}_q^{q^m} \\ & f & \mapsto & (f(x))_{x \in \mathbb{F}_q^m} \end{array}$$

Example: full length Reed-Solomon code ($n = q$).

$$\mathcal{C} = \mathrm{RS}_q(k) = \{\mathrm{ev}_1(f),\ f \in \mathbb{F}_q[X],\ \deg f < k\}$$

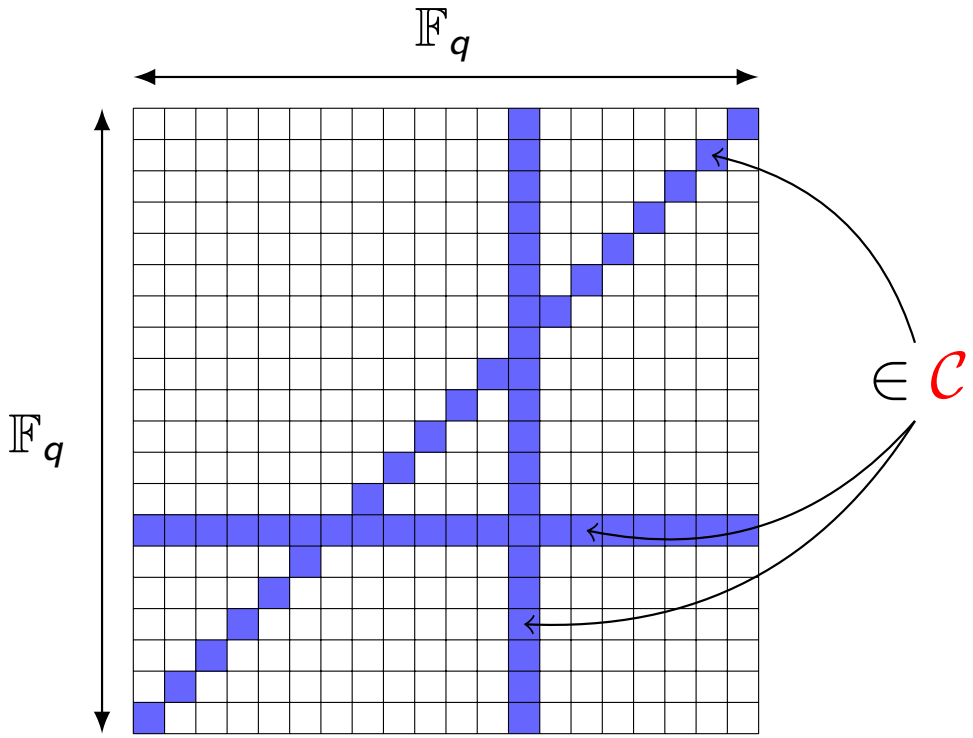


Affine lifting of codes

Let $\mathcal{C} \subseteq \{\mathrm{ev}(f), f \in \mathbb{F}_q[T]\}$ be a (univariate) *base code*.

The (multivariate) *lifted code* $\mathbf{Lift}_m(\mathcal{C})$ is the code:

$$\mathcal{L} = \mathbf{Lift}_m(\mathcal{C}) = \{\mathrm{ev}_m(g),\ g \in \mathbb{F}_q[X_1, \ldots, X_m],\ \forall \text{ affine line } \ell,\ \mathrm{ev}_1(g_{|\ell}) \in \mathcal{C}\}$$



Alan Guo, Swastik Kopparty, Madhu Sudan, *New Affine-Invariant Codes from Lifting* in Proceedings of ITCS'13, Berkeley, USA, 2013.

LiftPoR: Initialization and verification

Consider $\mathcal{L} = \mathbf{Lift}_m(\mathcal{C})$.

- Initialization:

$$\mathbf{Init}(F) = \begin{cases} \mathtt{data}_F & = \varnothing \\ \tilde{F} & = \mathrm{Enc}_{\mathcal{L}}(F) \end{cases}$$

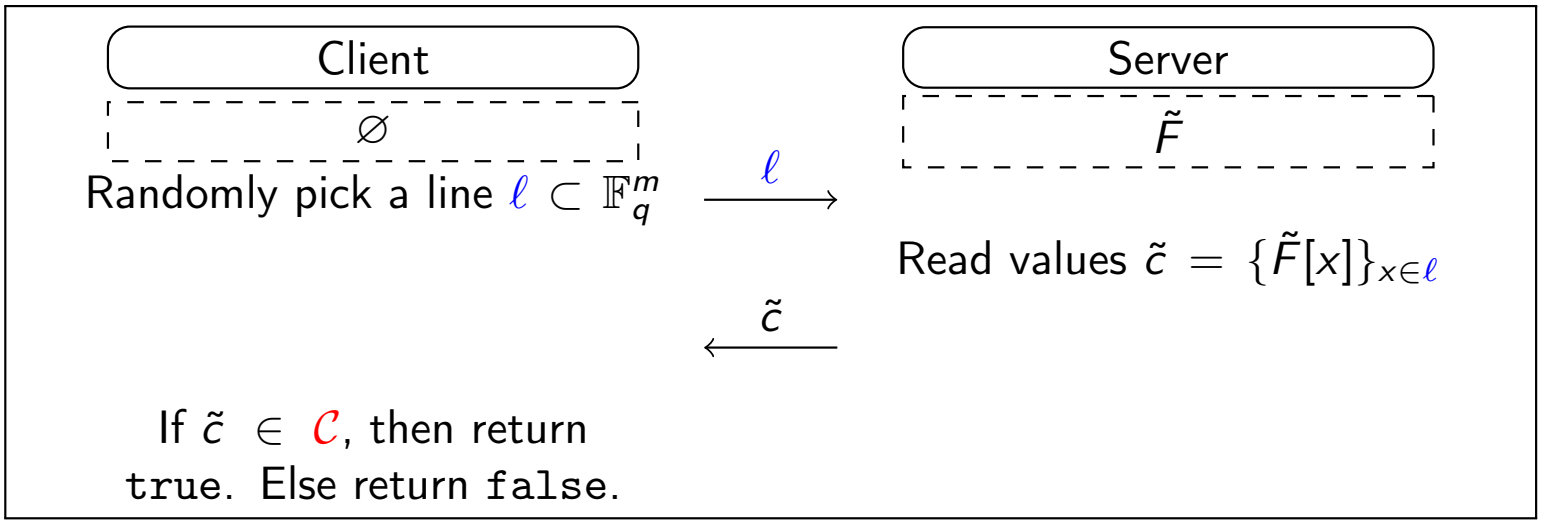
LiftPoR: Initialization and verification

Consider $\mathcal{L} = \mathbf{Lift}_m(\mathcal{C})$.

- Initialization:

$$\mathbf{Init}(F) = \begin{cases} \mathtt{data}_F & = \varnothing \\ \tilde{F} & = \mathrm{Enc}_{\mathcal{L}}(F) \end{cases}$$

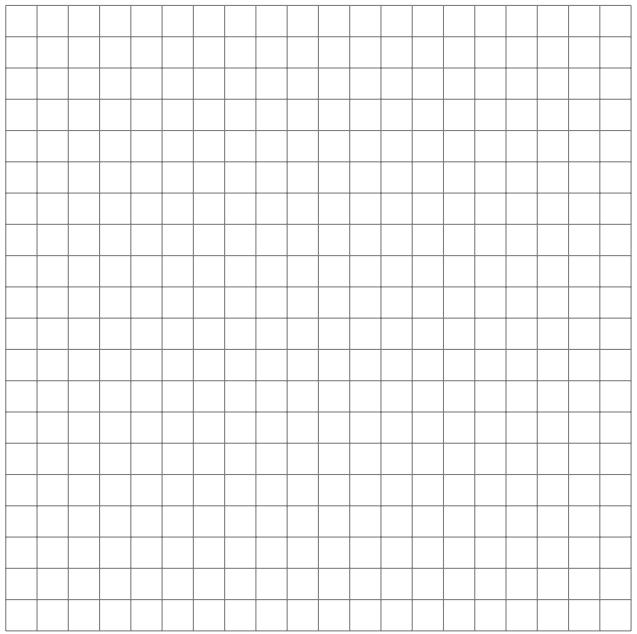
- Verification:



LiftPoR: Extraction

- Initialize file Tab with q^m erasures.

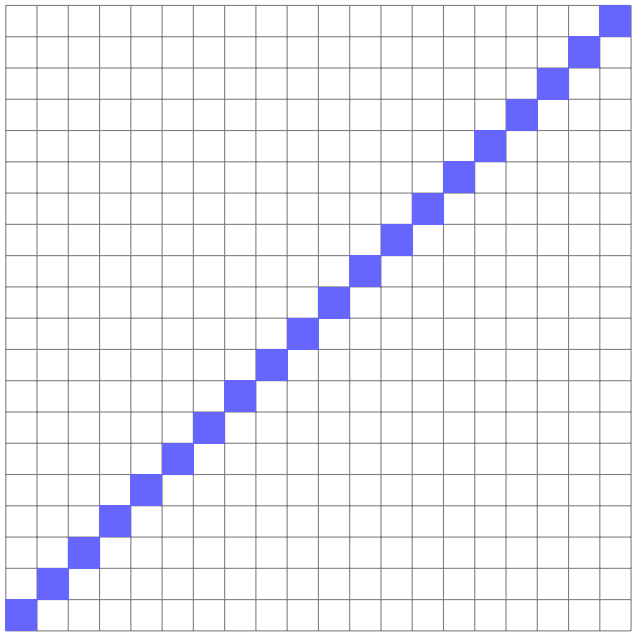
Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

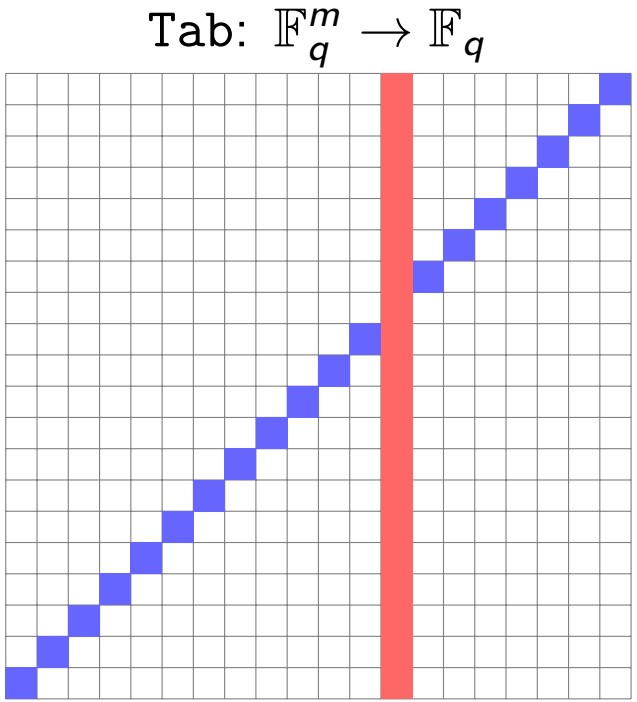
- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,

Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

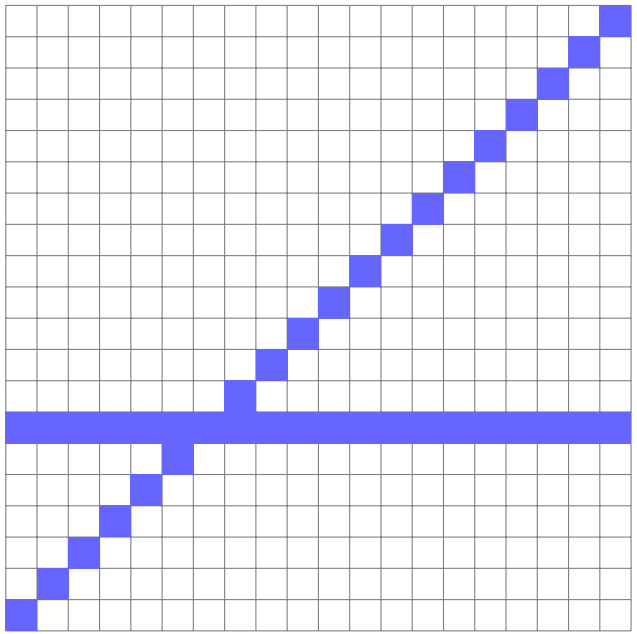
- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.



LiftPoR: Extraction

- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.

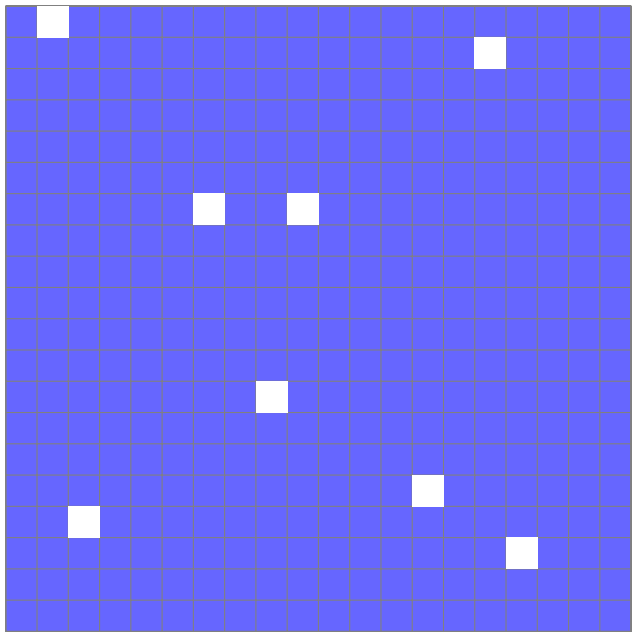
Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.

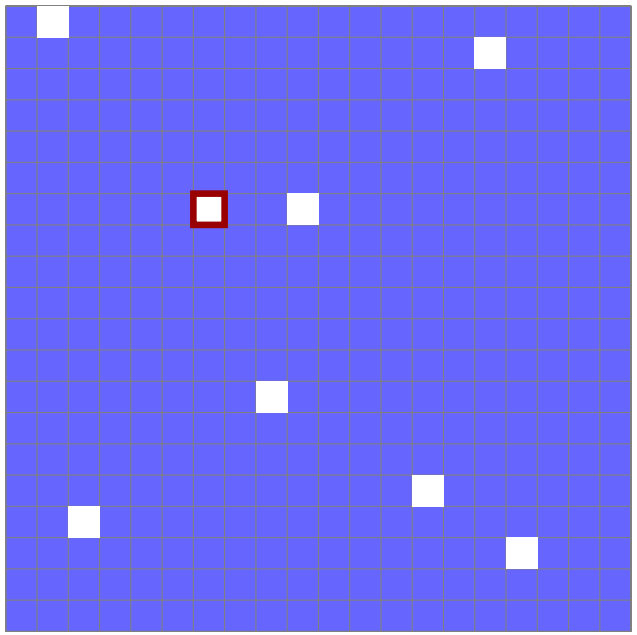
Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

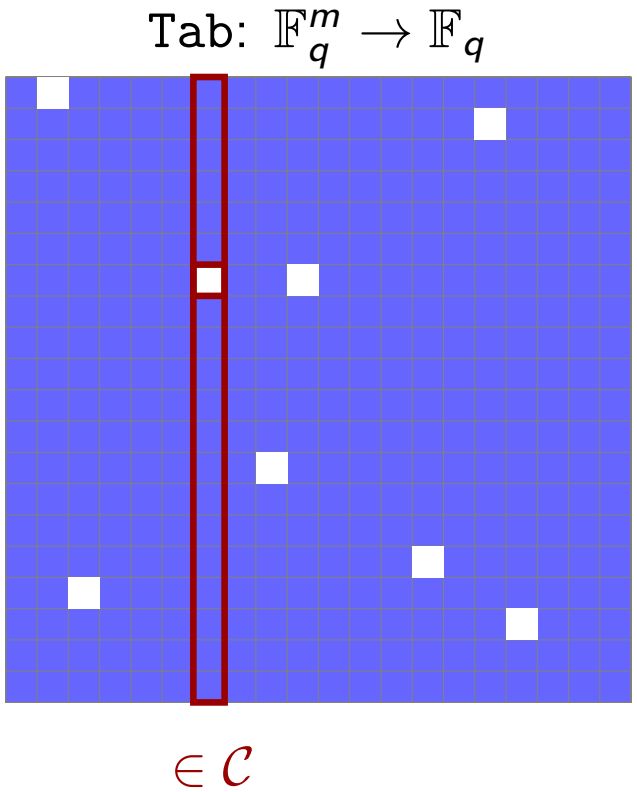
- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.
- Decode the remaining erasures with the decoding algorithm of $\mathcal{C}$.

Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

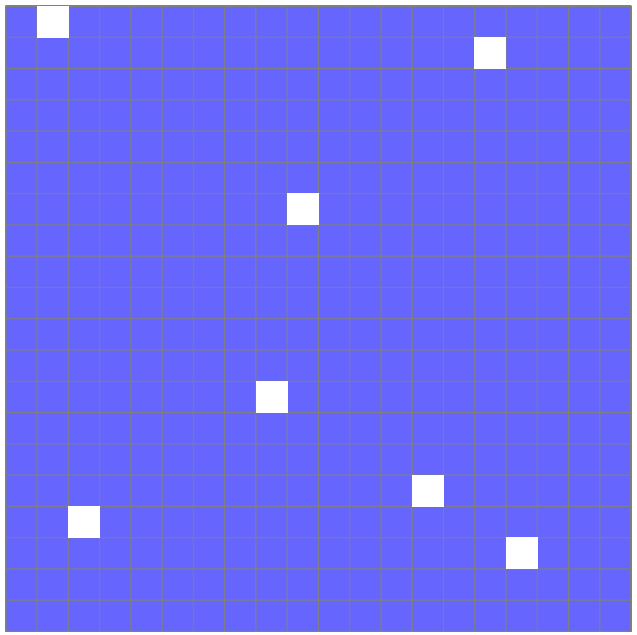
- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.
- Decode the remaining erasures with the decoding algorithm of $\mathcal{C}$.



LiftPoR: Extraction

- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.
- Decode the remaining erasures with the decoding algorithm of $\mathcal{C}$.

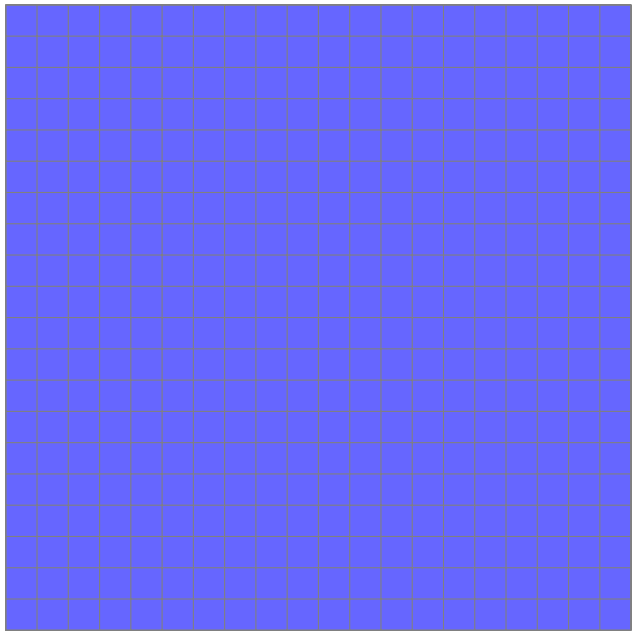
Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



LiftPoR: Extraction

- Initialize file Tab with q^m erasures.
- While there remains $\geq q^{m-1}$ erasures :
  - run a verification test;
  - if success: update the file,
  - otherwise, do nothing.
- Decode the remaining erasures with the decoding algorithm of $\mathcal{C}$.

Tab: $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$



Security analysis

Proposition. If the distance $d_{\min}(\mathcal{C}) \geq 2$, we can efficiently decode up to q^{m-1} erasures in $\mathbf{Lift}_m(\mathcal{C})$.

Let $A = \{ \text{ line } \ell \subset \mathbb{F}_q^m \}$.

Lemma. Let $U \subset A$ such that $|U| \geq \frac{1}{2} |A|$. Then, the union of lines in U covers all but q^{m-1} points of $\mathbb{F}_q^m$.

Security analysis

Proposition. If the distance $d_{\min}(\mathcal{C}) \geq 2$, we can efficiently decode up to q^{m-1} erasures in $\mathbf{Lift}_m(\mathcal{C})$.

Let $A = \{ \text{ line } \ell \subset \mathbb{F}_q^m \}$.

Lemma. Let $U \subset A$ such that $|U| \geq \frac{1}{2} |A|$. Then, the union of lines in U covers all but q^{m-1} points of $\mathbb{F}_q^m$.

Let $\mathcal{P}$ be a **LiftPoR**, with $m \geq 2$ and $d_{\min}(\mathcal{C}) \geq 2$. Then, for all $\tau < 1/2$, $\mathcal{P}$ is $(\tau, 1)$-sound.

Security analysis

Proposition. If the distance $d_{\min}(\mathcal{C}) \geq 2$, we can efficiently decode up to q^{m-1} erasures in $\mathbf{Lift}_m(\mathcal{C})$.

Let $A = \{ \text{ line } \ell \subset \mathbb{F}_q^m \}$.

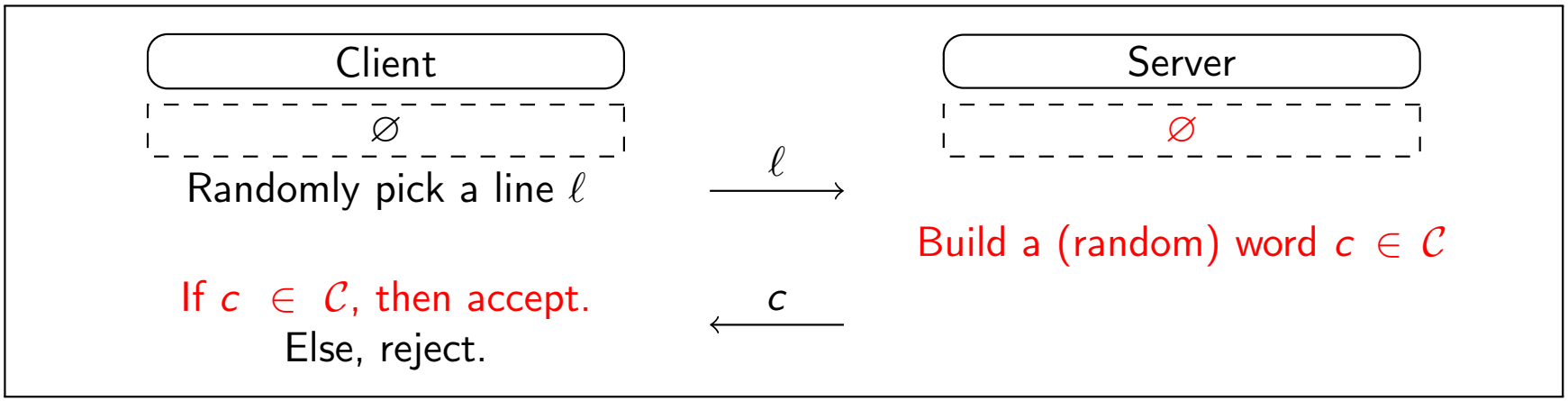
Lemma. Let $U \subset A$ such that $|U| \geq \frac{1}{2} |A|$. Then, the union of lines in U covers all but q^{m-1} points of $\mathbb{F}_q^m$.

Let $\mathcal{P}$ be a **LiftPoR**, with $m \geq 2$ and $d_{\min}(\mathcal{C}) \geq 2$. Then, for all $\tau < 1/2$, $\mathcal{P}$ is $(\tau, 1)$-sound.

Really?

An attack

Verification



Any word in $\mathcal{C}$ is accepted!

Improvement

How to prevent these **spurious codewords** with high probability?

Improvement

How to prevent these **spurious codewords** with high probability?

Idea: Link each symbol to its location in $\mathbb{F}_q^m$.
For any $x \in \mathbb{F}_q^m$, let $\phi_x : \mathbb{F}_q \to \mathbb{F}_q$ be a secret and invertible function.
The new $\tilde{F}$ is now defined by:

$$\tilde{F}_x = \phi_x(\mathrm{Enc}_{\mathcal{L}}(F)_x)$$

Improvement

How to prevent these **spurious codewords** with high probability?

Idea: Link each symbol to its location in $\mathbb{F}_q^m$.
For any $x \in \mathbb{F}_q^m$, let $\phi_x : \mathbb{F}_q \to \mathbb{F}_q$ be a secret and invertible function.
The new $\tilde{F}$ is now defined by:

$$\tilde{F}_x = \phi_x(\,\mathrm{Enc}_{\mathcal{L}}(F)_x)$$

In practice: a block cipher $\mathtt{E}_\kappa$ in counter mode (CTR) gives an appropriate keystream; *i.e.*:

$$\phi_x(\alpha) = \alpha \oplus \mathtt{E}_\kappa(x).$$

LiftPoR: the secure one

- Initialization, verification: just need to add the en(de)cryption step.
- Extraction:
  - Add a majority voting phase (with threshold γ) in order to increase the soundness.
  - Consequences: need more memory and time.

Theorem

Let $\mathcal{P}$ be the **LiftPoR** with parameters q, m, $d = d_{\min}(\mathcal{C}) \geq 2$ and $\gamma \in [0, 1]$.
Then, for all $\tau < (1 - \gamma)/2$, $\mathcal{P}$ is $(\tau, 1 - 2^{-\lambda})$-sound, where:

$$\lambda = \tilde{O}(\gamma q^{m-1} d)\,.$$

Instanciation and implementation

Numerical values:

Instance		Results				
m	q	$\lvert F\rvert$ (bits)	$1/R - 1$ (redundancy)	comm. (bits) Cl. $\rightarrow$ Se.	comm. (bits) Se. $\rightarrow$ Cl.	comm./$\lvert F\rvert$
2	64	$3,367$	0.217	24	384	0.121
2	$4,096$	$16,245,775$	0.033	48	$49,152$	0.0030
3	512	$79,837,411$	0.681	54	$4,608$	5.84×10^{-5}
4	128	$49,578,831$	4.414	56	896	1.92×10^{-5}

Implementation ($m = 2$, $d_{\min}(\mathcal{C}) = 2$) with SageMath (Python).

- Subquadratic time complexity for every phase, but ...
- ... high constants, e.g. extraction < 10 kbits/s, even without decryption.

Asymptotic results and comparison

$$
\begin{aligned}
|F| &: \text{file size} \\
|\kappa| &: \text{key size} \\
R = \frac{|F|}{q^m \log q} &: \text{code rate} \\
R^* = \frac{1}{R} - 1 &: \text{code redundancy}
\end{aligned}
$$

Work	**Bowers-Juels-Oprea** (2010, JK'07 improved)	**Shacham-Waters** (2008 – 13)	**Ours**
Unbounded use?	No (N uses)	Yes	Yes
Client storage	$\lvert\kappa\rvert$	$\lvert F\rvert^{\beta} + \lvert\kappa\rvert$	$\lvert\kappa\rvert$
Server storage	$\alpha N + R^*\lvert F\rvert$	$\frac{\lvert F\rvert^{\mathbf{1}-\beta}}{R} + R^*\lvert F\rvert$	$R^*\lvert F\rvert$
Communication cost	$\lvert\kappa\rvert + \alpha$	$\lvert F\rvert^{\beta} + \frac{\lvert\kappa\rvert}{R^*}$	$\left(\frac{\lvert F\rvert}{R}\right)^{\mathbf{1}/m}$
Notes	$\alpha \simeq 2^8$	$\beta \in]0,1[$	$m \in \{2,3\}$

Asymptotic results and comparison

$$
\begin{aligned}
|F| &: \text{file size} \\
|\kappa| &: \text{key size} \\
R = \frac{|F|}{q^m \log q} &: \text{code rate} \\
R^* = \frac{1}{R} - 1 &: \text{code redundancy}
\end{aligned}
$$

Work	**Bowers-Juels-Oprea** (2010, JK'07 improved)	**Shacham-Waters** (2008 – 13)	**Ours**
Unbounded use?	No (N uses)	Yes	Yes
Client storage	$\lvert\kappa\rvert$	$\lvert F\rvert^{\beta} + \lvert\kappa\rvert$	$\lvert\kappa\rvert$
Server storage	$\alpha N + R^*\lvert F\rvert$	$\frac{\lvert F\rvert^{\mathbf{1}-\beta}}{R} + R^*\lvert F\rvert$	$R^*\lvert F\rvert$
Communication cost	$\lvert\kappa\rvert + \alpha$	$\lvert F\rvert^{\beta} + \frac{\lvert\kappa\rvert}{R^*}$	$\left(\frac{\lvert F\rvert}{R}\right)^{\mathbf{1}/m}$
Notes	$\alpha \simeq 2^{\mathbf{8}}$	$\beta \in]0,1[$	$m \in \{2,3\}$

Conclusion

- We build (theoretically) efficient proofs of retrievability which check the structure of the encoded file, with:
  - low storage (especially for the client);
  - quite low communication;
  - is implementable...
- Open questions/future works:
  - other locally decodable/testable codes;
  - link with Private Information Retrieval schemes...