DESIGNING CRYPTOGRAPHY FOR SMALL ORGANIZATIONS AND PROJECTS

HTTPS://ARXIV.ORG/ABS/2109.10074 / HTTPS://EPRINT.IACR.ORG/2022/981

RWC2023

FrodoPIR

Sofía Celi (Brave Software), Alex Davidson (Universidade Nova de Lisboa), Pete Snyder (Brave Software)

- This talk does not -heavily- deal with theoretical concerns
- It concerns about *who benefits from cryptography* and *what kind of systems we decide to deploy and release*

The limits of my language are the limits of my world

-Ludwig Wittgenstein

Cryptography in practice

Does not work for the many:

- Do not take into account many cases:
  - Authentication systems do not take into account device seizure or coercion in IPV cases, for example (to cite a few)
- Has high computational and communication costs that are unfeasible to be deployed by **small organizations and projects**
- Has high financial costs, requires specific hardware (that can be unavailable on certain regions of the world), or “good” Internet connectivity
- It is not concerned with policy or legal regulations
- It is not concerned with human-rights considerations
- Contributes to a centralized Internet
- Lacks implementations or its implementations are not usable
- **Becomes privacy and security for the ones that can afford it**

- Big organizations have been pushing for new privacy-preserving schemes:
  - This is great! (...)
  - But does not work for the many

- Two examples:
  - Privacy-preserving measurement (PPM) schemes
  - Private information retrieval (PIR) schemes

Can these proposals be used by small organizations that have constrained budgets or are volunteer run?

The proposed schemes (for PPM or PIR) are:

a. Computationally and communicationally expensive
b. Rely on non-collusion assumptions for servers/clients that communicate with each other
c. Require several rounds of communication
d. Need to trust third parties or to have specialized hardware
e. Require high financial costs
f. Use “complex” cryptography that might not be available for everyone and can be expensive
g. Do not work for many data types and require maintaining different codebases

- It is great to have schemes that provide privacy and can be used in practice if you can afford them...
- But:
  a. Are these the ***only schemes*** we choose to standardise and deploy?
  b. Do they **work for everyone and in many cases**?
  c. Are we forgetting the people that will use them? Do we only care about people that look like us as our “end-user”? Is this the limit of our world?
  d. Are we denying access to systems due to the technology we push forward?

Is this the technology we chose to create?

“Si tú vas en verdad porque estás mal... si estamos hablando como de derechos sociales como salud, estar poniendo más encima un prerrequisito que es la huella, lo encuentro como super elitista. . . hay un montón de gente que trabaja, hace trabajo manual.. y efectivamente sus huellas digitales no deben ser las más claras posibles, entonces como que creo que establecer un sistema biométrico solo habla de un grupo de personas súper específico...”

(“If you go it’s because you're not well... if we're talking about social rights like health, and you are adding a prerequisite that is the biometric, I find you super elitist. . . there are a lot of people who work, they do manual work... and truly their biometrics will not be as clear as possible, so I believe that using a biometric system only works for a super specific group of people...”)

Narrativas en torno al uso de la huella digital en la salud pública -
Javiera Figueroa & Catalina Venegas, Derechos Digitales

- Authentication is great!
  - But, some of the systems we have in practice don’t work for the many
  - In fact, they work for the few!
  - But, they deny access to other services:
    - Access to health services
    - Access to education
    - Access to economic transactions

Do some of these privacy-preserving schemes fall in the same trap?

- If you cannot run an expensive scheme, does it mean that you cannot access a DB (that holds your medical records, for example) or that your usage of a system is not taken into account (so it is not designed for you)?

- In the case of new deployed (or to be) privacy-preserving schemes:
  - Do they work only for the ones that can afford it?
  - Do they limit the access to further services?
  - Do they really push for privacy as a human right?

- We need PPM:
  - Software/Hardware metrics are essential
- We need PIR:
  - Because private information retrieval to a database is essential

Let’s look at one limitation

Non-colluding servers:

- Where can these servers be located if different regions of the world have different policies/legislations around private data handling?
- How we can generate trust on this assumption?
- What is their legal/contractual model?
- What is their economic model? Is it “pay-for-usage”?

This is not a theoretical concern:

Teams building software for privacy-critical users have highly-constrained budgets, or may even be volunteer-run.

Cryptographic systems that require large budgets will not serve many categories of users

(despite the important and promising contributions of these high-resource-requirement systems).

Change of priorities

Smaller organizations may prioritize:

- Simpler trust assumptions and/or requirements
- Simpler (or more established) cryptographic primitives
- Lower cost
- Better failure modes

We constructed to schemes following this approach (note, though, that they are not perfect, they have limitations)

STAR

https://arxiv.org/abs/2109.10074 /
https://brave.com/research/star-secret-sharing-for-private-threshold-aggregation-reporting/

ACM CCS 2022

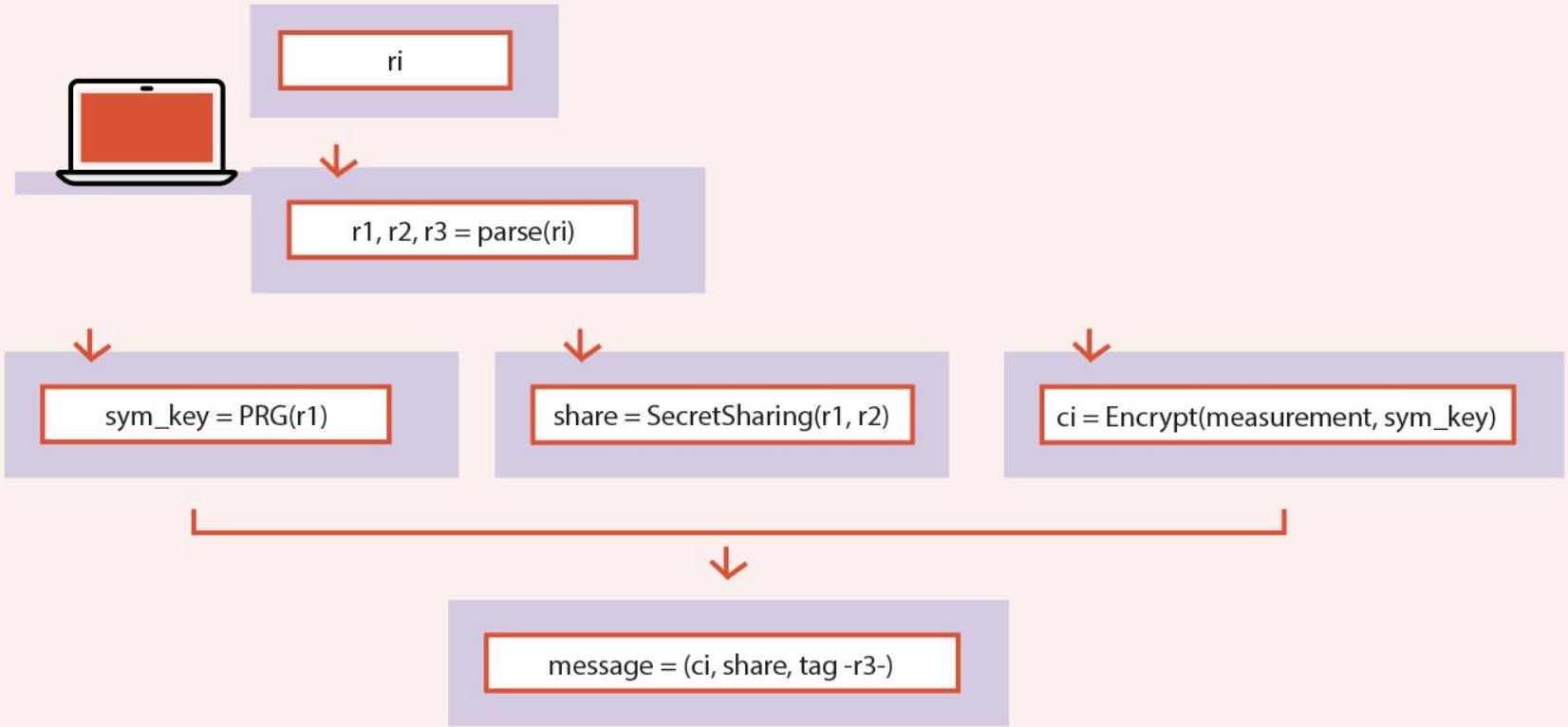
STAR

- A *K-Threshold Aggregation* scheme that uses:
  - OPRFs for deterministic randomness derivation (this can be derived locally)
  - *Adept Secret Sharing*
  - Encryption algorithm
- Works with any data type, but has a *degree of leakage*

Three phases:

1. Randomness extraction (either local or with a OPRF server)
2. Local client share creation
3. Server aggregation and reveal

Share creation



Aggregation and reveal phase

Groups together messages with the same tag into a set

Divides the set into subsets of K size

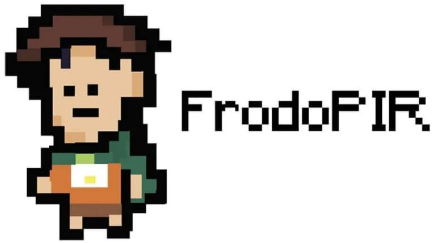
r1 = recover(share)

sym_key = PRG(r1)

measurement = decrypt(ci, sym_key)

- Easy to implement
- Uses “boring” cryptography
- No need for trust assumptions of non-collusion or trusted hardware
- The total costs of running all the components in STAR(*) are $0.00409 + $0.037 + $0.0053 = $0.04639, which is more than 24× cheaper than the cost of running the nearest scheme ($1.1152).

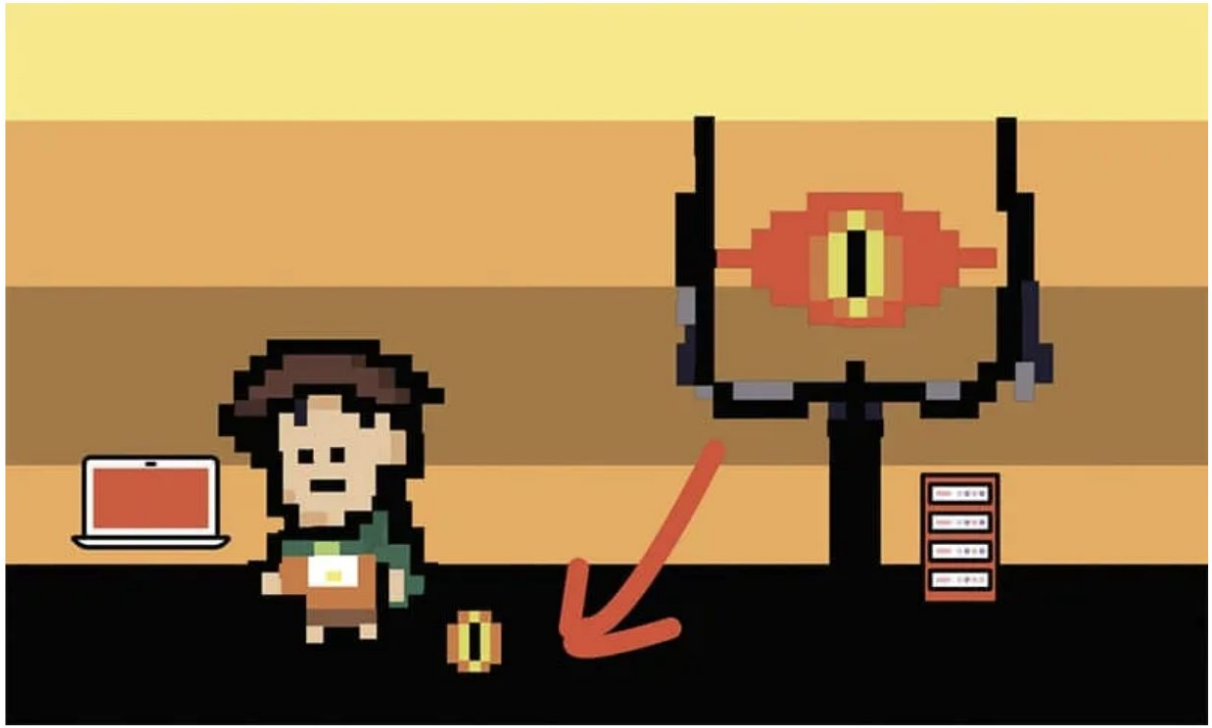
(*) On AWS EC2 c4.8xlarge at the time of writing



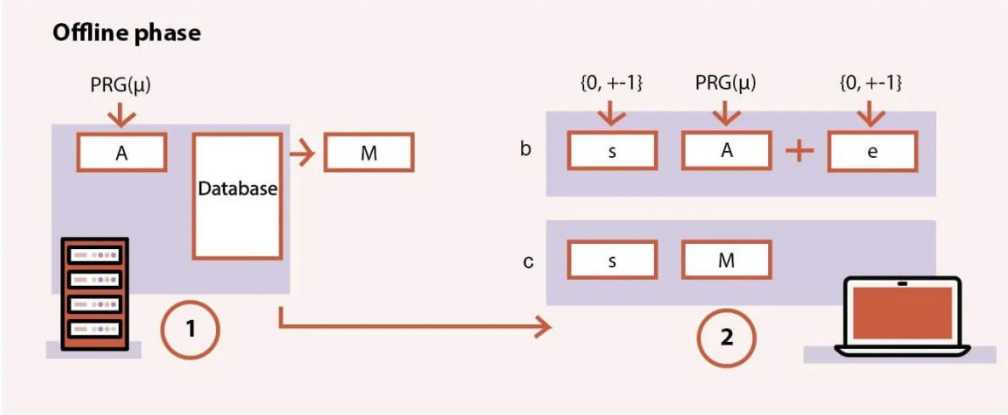
FRODOPIR

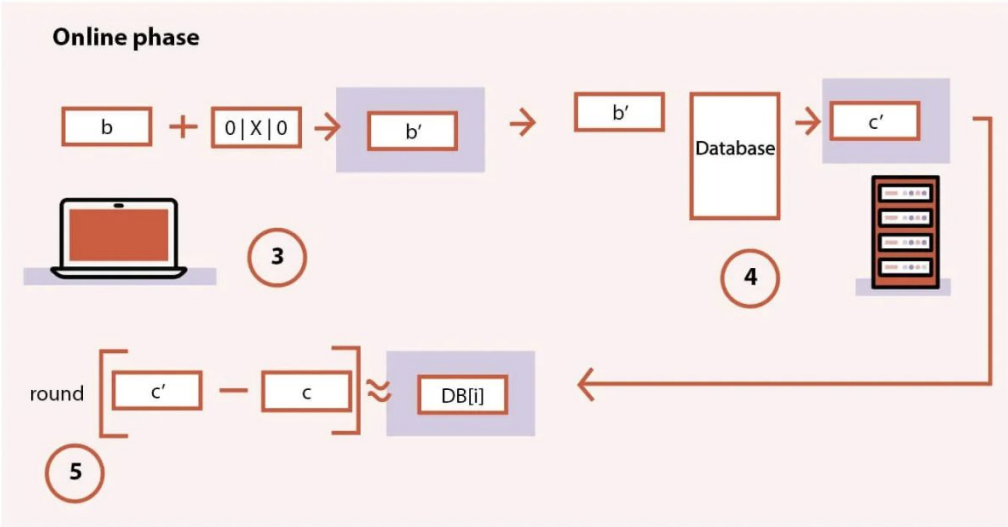
https://eprint.iacr.org/2022/981 /
https://brave.com/frodopir/
TO APPEAR AT POPETS 2023

Announcing FrodoPIR!



Just as the state of Sauron (its ring) moved to Frodo, we can move the mu and A to the client. The client then can then perform hidden queries to the server, just as Frodo remains hidden from Sauron.


Offline phase
PRG(μ)
A
Database
M
1
{0, +-1}
PRG(μ)
{0, +-1}
b
s
A
e
c
s
M
2


Online phase
b
0 | X | 0
b'
3
b'
Database
c'
4
round
c'
c
DB[i]
5

- LWE-based PIR schemes do not require the server to store any extra per-client state
- LWE-based PIR schemes are faster and cheaper
- LWE-based PIR schemes are simple to implement: they require no polynomial arithmetic or fast Fourier transforms
- Scheme fits all of the operations in 32-bit integers
- It does not require any non-colluding server
- It has explicit design decisions: downloading “offline” parameters takes time but it is done sparsely

Why it works for small organizations?

- Usage of “boring” cryptography → cheap costs and fast times
- One single-server model:
  - We don’t need to think on non-collusion
  - We don’t need to consider legal implications
- We don’t need to buy specialized hardware
- We can easily implement and maintain

FUTURE

Inspiration from what we saw

- “Boring” cryptography does work, even when it is not “novel enough”
  - It is available for many
  - It is cheap
  - It is fast and with low computational costs
  - It still has lots of open (*and novel!*) problems
- “Simple” trust assumptions work:
  - A third-party non-colluding server is hard to get by
- We need to address financial costs as part of our research
- Bandwidth costs are a real concern

- We need to think as community outside of the limits of our worlds
- Privacy is crucial but it should not only be for the ones that can afford it
- We should create systems that do not deny access to other services

The limits of our world need to be expanded:

- Inclusive cryptographic design for diverse people
- Provide privacy/security properties for the many (even when the schemes that are usable for the many are not “novel enough”)
- Provide schemes that aim to enhance access rather than deny it

- It is not easy, but together we have the ability of doing it

References

- STAR:
  - Paper version for ACM CCS by Alex Davidson, Peter Snyder, E. B. Quirk, Joseph Genereux, Benjamin Livshits and Hamed Haddadi: https://dl.acm.org/doi/abs/10.1145/3548606.3560631
  - Code: https://github.com/brave/sta-rs
- FrodoPIR:
  - Paper version of FrodoPIR to appear at PETS2023 by Sofía Celi, Alex Davidson and Gonçalo Pestana: https://www.petsymposium.org/popets/2023/popets-2023-0022.pdf
  - Code: https://github.com/brave-experiments/frodo-pir
  - Blog post: https://brave.com/frodopir/
  - “One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval” by Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn and Vinod Vaikuntanathan: https://eprint.iacr.org/2022/949.pdf

THANK YOU!

@claucece
The Brave Research Team

YOU HAVE UNLOCKED SECRET SLIDES!

Protocol	Single-round interaction with clients	Bandwidth	Client computation	Aggregation computation	Single-server aggregation	Associated data	Negligible correctness errors	Fail-safety
Proxy-based shuffling [8, 13, 31]	✓	$O(n)$	$O(1)$	$O(n)$	✗	✓	✓	✗
Kissner et al. [27]	✗	$O(mn\lambda)$	$O(n^2)$	$O(mn\lambda)$	✓	✗	✓	✗
Blanton et al. [9]	✓	$O(mn\lambda)$	$O(n^2)$	$O(mn^2\lambda)$	✗	✗	✓	✗
Randomized response [4, 5, 12, 34, 40]	✗	$O(n\lambda)$	$O(1)$	$O(n)$	✓	✗	✗	✓
Boneh et al. [10]	✓	$O(mn\lambda)$	$O(\lambda)$	$O(mn\lambda\kappa)$	✗	✗	✓	✗
STAR (Section 4)	✓	$O(n\lambda)$	$O(\lambda)$	$O(n\lambda\kappa^2)$	✓	✓	✓	✓

Coarse-grained comparison of STAR against previous work. We use λ to denote the security parameter, $n = |C|$ to denote the number of clients, and m to denote the number of servers that are used in multi-server settings. Note that we ignore generic MPC techniques for computing threshold aggregation due to well-established performance limitations. We also do not include Prio-like protocols as they are not compatible with string-based data.

Approach	Security assumptions	Client costs					Server costs				
		Communication		Computation		Storage	Communication		Computation		Financial
		Offline	Online	Offline	Online		Offline	Online	Offline	Online	
Stateless [6, 62, 3]	RLWE	—	m	—	m	—	—	1	—	m	$\$5.2 \times 10^{-3}$
PSIR [65]	RLWE	—	1	m	$\sqrt{m}$	$\sqrt{m}$	$\lvert DB \rvert / \sqrt{m}$	1	—	m	$\$8.8 \times 10^{-5}$
SOnionPIR [62]	RLWE	$\sqrt{m}$	1	$k \cdot \sqrt{m}$	k	$\sqrt{m}$	$\sqrt{m}$	1	$\sqrt{m}$	m	$\$6.4 \times 10^{-4}$
CHKPIR [30]	RLWE	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	1	$\sqrt{m}$	$\sqrt{m}$	$\sim \$8.8 \times 10^{-5\dagger}$
FrodoPIR	LWE	—	m	m	1	λ	$\lambda \cdot m^{-1/2}$	1	$\sqrt{m}/C$	m	$\$(1.9/C \times 10^{-2} + 1.3 \times 10^{-5})$

Asymptotic comparison focusing on the dependency on the number of database elements, m, of practical approaches for realizing single-server PIR with adaptive queries (i.e. not including batch PIR schemes, logarithmic factors are ignored). All costs are amortized according to C clients that launch c = √ m queries. Communication costs relate to the amount of data that is sent to the party. The financial costs are given relative to a database containing 2^{20} 1KB elements, are amortized per-query and per-client. The costs of CHKPIR are assumed to be zero for the online phase, and are thus completely dominated by the offline phase.

STAR assumption on an OPRF server

- A OPRF server is used for generation of deterministic randomness:
  - Cheap process
  - Multiple servers that are already used for this are available
  - No computation of private data

- Local functionality can be used as well