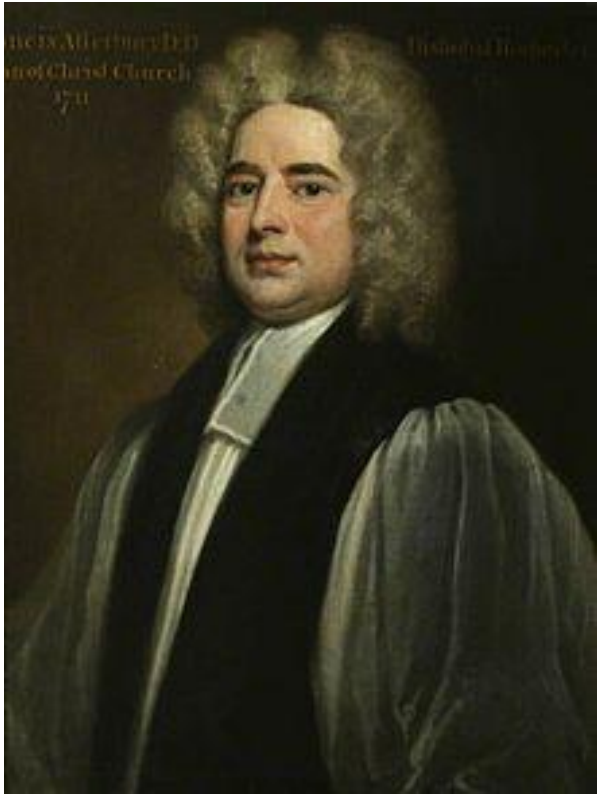
ZKP and the Law

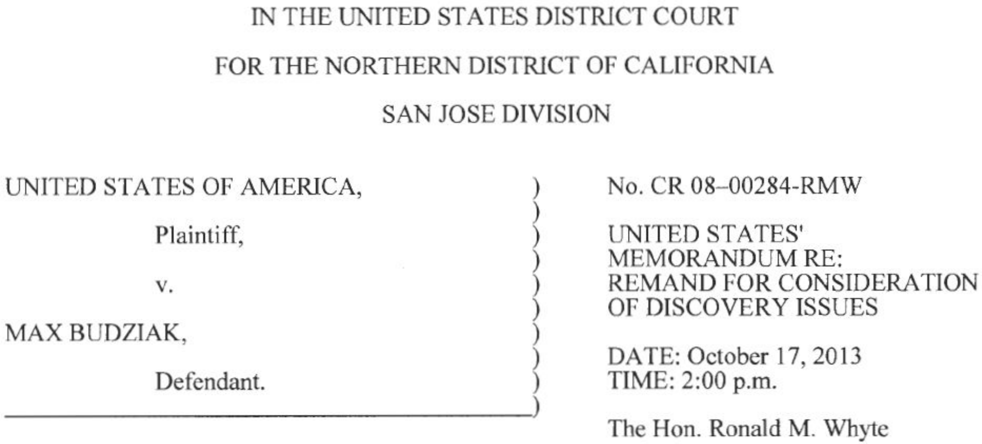
Ran Canetti
Boston University

Based on:

Kenneth Bamberger, -, Shafi Goldwasser, Rebecca Wexler, Evan Zimmerman: Verification Dilemmas in Law and the Promise of Zero-Knowledge Proofs. Berkeley Technology Law Journal, Vol. 37, No. 1 (2022).

Dor Bitan, -, Shafi Goldwasser, Rebecca Wexler: Using Zero-Knowledge to Reconcile Law Enforcement Secrecy and Fair Trial Rights in Criminal Cases. SSRN (2022).


ncis Atterbury DD
an of Christ Church
1711

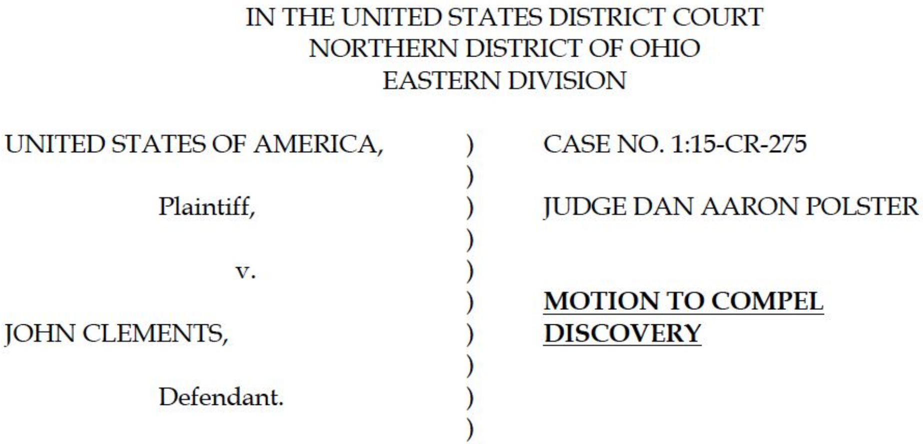


IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

UNITED STATES OF AMERICA,	)	No. CR 08–00284-RMW
	)	
Plaintiff,	)	UNITED STATES' MEMORANDUM RE: REMAND FOR CONSIDERATION OF DISCOVERY ISSUES
v.	)	
	)	
MAX BUDZIAK,	)	
	)	DATE: October 17, 2013 TIME: 2:00 p.m.
Defendant.	)	
	)	The Hon. Ronald M. Whyte



IN THE UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF OHIO
EASTERN DIVISION

UNITED STATES OF AMERICA,	)	CASE NO. 1:15-CR-275
	)	
Plaintiff,	)	JUDGE DAN AARON POLSTER
	)	
v.	)	
	)	**MOTION TO COMPEL**
JOHN CLEMENTS,	)	**DISCOVERY**
	)	
Defendant.	)	
	)	

Verification dilemmas in criminal justice: hiding investigation tools vs. due process

U.S. v. Budziak:

> “A party seeking to impeach the reliability of computer evidence should have sufficient opportunity to ascertain by pretrial discovery whether both the machine and those who supply it with data input and information have performed their tasks accurately.”
> United States v. Liebert, 519 F.2d 542, 547–48 (3d Cir. 1975);
>
> “It is quite incomprehensible that the prosecution should tender a witness to state the results of a computer’s operations without having the program available for defense scrutiny and use on cross-examination if desired.”
> United States v. Dioguardi, 428 F.2d 1033, 1038 (2d Cir.1970)

- On appeal, 9th circuit determined that defendant should have been allowed to inspect the program, and remanded to district court, who ordered FBI to disclose the software.
- FBI were unable to locate the software, the case was dismissed and the defendant released from jail.

A counterpoint to Budziak: US. V. Clements

- FBI procured the modified version of the P2P software from an external vendor
- Files downloaded by the hidden software were the only evidence for indictment
- After initial motions sides agreed to use an independent expert that would inspect source and executable and respond to agreed-upon yes/no questions. Eventually, expert obtained access only to source but still completed a report, answering all questions favoring LE.
- Defense expert argued that (a) indep. Expert could not have responded without executable and (b) the evidence found mandated the opposite response to binary questions. However judge accepted evidence and the case ended in a plea bargain.

P2P software

- Users share large files directly, no central server

  Each file is split among multiple users

  Users collect pieces, assemble locally

   more robust, enhances privacy

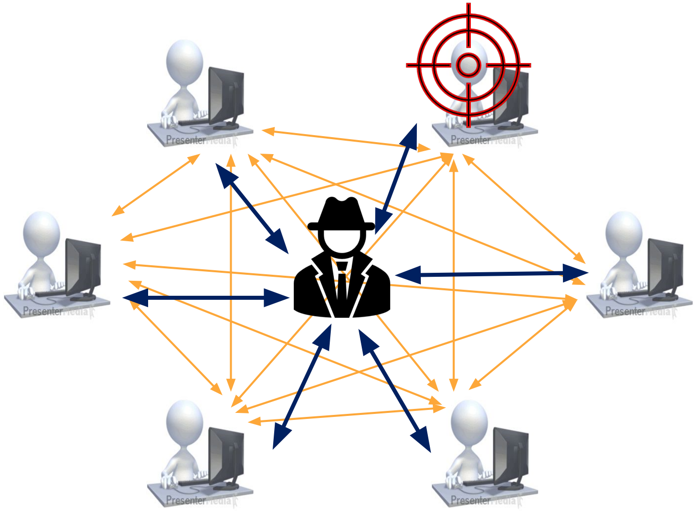
- Unfort, used also for sharing CSAM.

- Law enforcement investigates:

  Needs “single source download” evidence

   Can’t use the same software:

   Use a modified version of the program.



- ! In court: use ZKP to prove in court that “the modified version is legitimate.”

Challenges posed: [U.S. v. Budziak, U.S. V. Clements]

Technical questions:

- Did the software really get all pieces of the file from the target computer?

- Did the software have the capability to change “sharing permissions” of files on target computer? (specific exploits cited)

Translation to Legalese (questions to indep. Expert in US v. Clements):

Assertion 1. “The law enforcement software does not exploit a vulnerability in any software.”
Assertion 2. “The law enforcement software does not misuse the protocols on the [P2P] network to do things that it was not intended to do.”
Assertion 3. “The law enforcement software does not change any configuration options in other users’ software to extract information that would normally not be publicly available (e.g. it does not make other users’ software share private parts of their file systems

Our contributions

The Legal Framework

Using ZKPs: four-step legal workflow

- Objective: Construct a legal setting for ZKP to step in.
- We suggest: The four-step legal workflow.
  - Starting point: the claims relate to a secret object **X**.
  - In this talk (and our implementation), **X** is a text file.
- The main idea:
  - Have a check program **C** that checks **X**.
  - Write **C** in a ZKP-friendly programming language.
- The legal workflow:

$$C_D(X) = \begin{cases} 1 & \text{The claims are true} \\ 0 & \text{Otherwise} \end{cases}$$

$$C_D(X,R) = \begin{cases} 1 & \text{The claims are true and } Hash(X,R) = D \\ 0 & \text{Otherwise} \end{cases}$$

I know X for which $C_D(X)=1$

I know X,R for which $C_D(X,R)=1$

If program is bought from a private vendor switch roles

1. Commit

2. Design initial check program

3. Negotiate more checks

4. Run ZKP

1. Law enforcement publishes $D = Hash(X,R)$ (in advance).
2. When submitting **X**-based evidence, the prosecution submits an initial design of *C*.
   - ***C*** indeed corroborates that the claims are true.
   - The fact ***C_D(X,R)=1*** does not compromise secrecy of **X**.
3. The defense (expert witness) inspects ***C***.
   - If disagree, request an alternative ***C***.
   - If law enforcement again disagree, adjudicate ***C*** before the judge.
4. ZKP: Law enforcement runs **ZKP-prove(C,D,X) = P**, Defense (or court) runs **ZKP-verify(C,D,P) = Accept / Reject**.

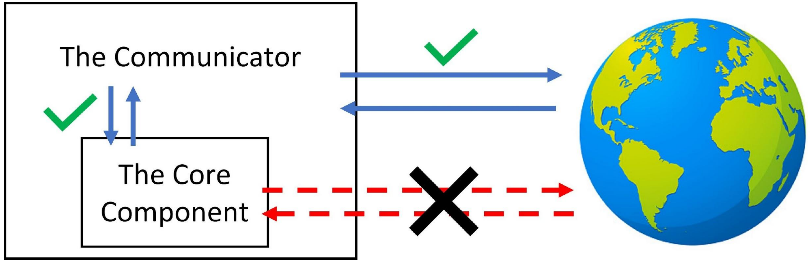
The proof of concept ZKP

Building the check program

1. The modified version only sends download requests generated via the original, public P2P software.
2. The modified version logs interactions accurately.

- Step 1: Restrict.
  - Instead of checking the entire program
  - Impose a structural restriction on the program:

  The split architecture
- Step 2: Publish non-secret part.
  - The Communicator is published.
  - The check program **C** checks the claims on the source code of the core component.
- Step 3: Check program via text processing of source code.
  - Check that the core contains a call to the Communicator.
  - Check that the core does not contain keywords, library names, parameters, or programming commands essential for writing programs that enable connecting to the Internet.
- Not foolproof!

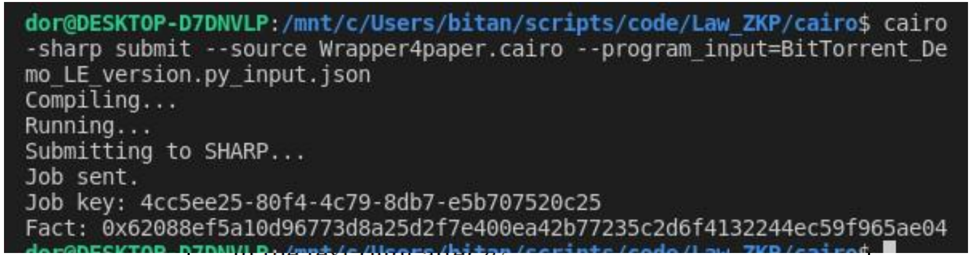


Not Sandboxing

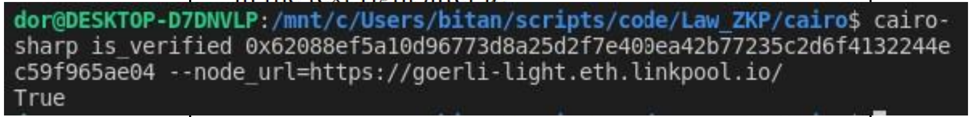
Would provide a meaningful level of assurance

Implementing the check program in Cairo

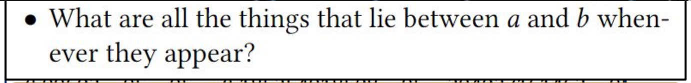
- Cairo does not support strings natively □
- The Initializer (coded in Python)
  - ✔ Convert text to Cairo readable format
  - ✔ Cryptographic commitment
- Cairo text-processing tools
- Cairo check program (use Cairo built-in for hash)
- [BBHR'19, GPR'21]: When writing in Cairo, can ZKP:
- "I know a secret input ***X*** for Cairo program ***C*** s.t. ***C(X)=1***"
- We did not implement local proof generation and verification.
- Instead, we used StarkWare's online services □
  - SHARP (current implementation is not in a version that supports ZK)
  - Verifier (smart contract)



```
dor@DESKTOP-D7DNVLP:/mnt/c/Users/bitan/scripts/code/Law_ZKP/cairo$ cairo
-sharp submit --source Wrapper4paper.cairo --program_input=BitTorrent_De
mo_LE_version.py_input.json
Compiling...
Running...
Submitting to SHARP...
Job sent.
Job key: 4cc5ee25-80f4-4c79-8db7-e5b707520c25
Fact: 0x62088ef5a10d96773d8a25d2f7e400ea42b77235c2d6f4132244ec59f965ae04
```



```
dor@DESKTOP-D7DNVLP:/mnt/c/Users/bitan/scripts/code/Law_ZKP/cairo$ cairo-
sharp is_verified 0x62088ef5a10d96773d8a25d2f7e400ea42b77235c2d6f4132244e
c59f965ae04 --node_url=https://goerli-light.eth.linkpool.io/
True
```



- What are all the things that lie between *a* and *b* whenever they appear?

Also: Compute Bag-of-words.

D := Hash(X,R) = 155730460542225362….

Low level language

No loops

Immutable memory

Choices and justifications

- Yes, law enforcement can cheat, but
  - Requires advanced planning and establishes deliberate malfeasance that raises the stakes of getting caught.
  - Requires substantially more efforts than after-the-fact lying in an affidavit.
  - The risk of lying about the program used already exists ($X' \neq X$) in the current legal baseline.
- More generally – knowing that ZKP solutions are feasible, courts could require them.

From technological feasibility to common practice

Current process for evaluating evidentiary privilege of law enforcement:

1. Is the requested information material to the defense?
2. Can the information be obtained in alternative means?
3. Has LE made reasonable accommodations to make the information as available as possible?

How should the process change in light of the technological feasibility of ZKPs?

- If LE is using ZKPs: can apply the same process, but on the new facts
- If LE is not using ZKPs, even though they are feasible and effective ?

   We propose to consider this as violation of (3)

Verification dilemmas in other legal settings:

- Regulatory actions:
  keeping audits unpredictable *and* demonstrably even-handed
- Justify administrative actions based on population studies without violating people’s privacy
- mergers and acquisitions:
  Impressing potential buyer vs. leaking secrets to competitors
- Trade secret litigation:
  demonstrate properties of data or algorithms that are trade secret, without exposing them
- …

Conclusion

- The concept of zero Knowledge can reshape established forms of human interaction, eliminating “inherent” caveats.
- We should keep our eyes out to catch these and disrupt!