(AD)SNARK: Improving its Scalability for Larger Secure Applications

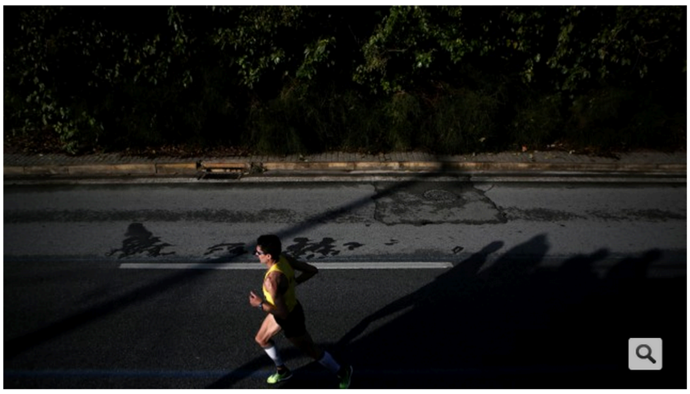
Manuel Barbosa
High Assurance Software Laboratory
INESC TEC and Minho University

Portugal

CPD–MEI, December 2014

21. November 2014, 07:19 Neues Krankenversicherungsmodell

Generali erfindet den elektronischen Patienten

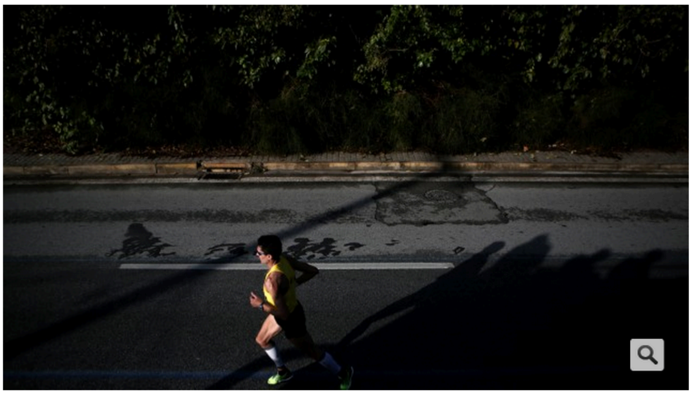


Wer Sport treibt, profitiert. Generali-Kunden müssen ihre Daten dazu aber per App an den Versicherer übermitteln. (Foto: AFP)

- Als erster großer Versicherer in Europa setzt die Generali-Gruppe künftig auf die elektronische Kontrolle von Fitness, Ernährung und Lebensstil.
- Kunden werden Gutscheine und Rabatte bei Prämien gewährt, wenn sie gesund leben. Dazu übermitteln sie der Generali über eine App regelmäßig Daten zum Lebensstil.
- Das Kalkül dabei: Wer gesund lebt, kostet den Krankenversicherern weniger Geld. Im Gegenzug erhalten willige Verbraucher Vergünstigungen.
- Aus Datenschutzgründen ist das neue Modell problematisch.

November 21, 2014, 07:19 New health insurance model

Generali invents the electronic patient



To play sport, benefited. Generali customers need but to transmit their data via app to the insurer. (Photo: AFP)

- As the first major insurer in Europe, the Generali Group has opted for the electronic control of fitness, nutrition and lifestyle.
- Customers are coupons and discounts on premiums paid when they live healthy. For this purpose, they shall provide the Generali a regular app data lifestyle.
- The calculus here: Who live healthy, costing the health insurers less money. In turn, consumers willing to receive benefits.
- For privacy reasons, the new model is problematic.

Main Reference

- ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data
- Joint work with M. Backes, Dario Fiore and R. Reischuk
- Paper available on ePrint

Concept

Three-party application scenario:

- Data Owner (*wants privacy*)
  - Wishes to keep her data x secret.
  - Must reveal partial information $f(x)$ to a Service Provider.
- Service Provider (wants integrity modulo legitimacy)
  - Does not trust Data Owner to correctly compute $f(x)$.
  - Will only provide service if convinced of $f(x)$'s correctness.
- Trusted Source (*defines legitimacy*)
  - Authenticates x (could even be the source of x).
  - Is trusted by Data Owner not to reveal x.
  - Is trusted by Service Provider to vouch for the legitimacy of x.

Data owner should be able to create a proof s.t.

- Service Provider is convinced that $f(x)$ was computed correctly,
- But it does not really learn anything about x;
- Except that x was authenticated by the Trusted Source.

Example #1: Health Risk Assessment

Just around the corner (c.f. Generali – Germany):

- Wearable biosensor collects your health information;
- If you give this information to your health insurance company;
- Then risk assessment could lower your premium.

Maps to three-party model:

- You are the Data Owner and want to keep your data private.
- Insurance company is the Service Provider.
- Biosensor can be Trusted Source if:
  - Cryptographically authenticates measurements.
  - Cannot be tampered with.

You compute premium yourself and provide proof that it was computed on authenticated readings.

Example #2: Smart Metering

Pupular example for privacy preserving data processing:

- Commodity provider installs trusted meter in your house;
- Trusted meter periodically collects readings, which you collect;
- Readings can reveal personal information; still ...
- You need to convince provider that you are paying for your consumption.

Maps to three-party model:

- You are the Data Owner and want to keep measurements private.
- Commodity provider is the Service Provider.
- (Smart) meter can be Trusted Source if:
  - Cryptographically authenticates measurements.
  - Cannot be tampered with.

You compute bill yourself and provide proof that it was computed on authenticated measurements.

Example #3: Financial audits

For the people with the money to pay for this technology:

- Your company/bank keeps extensive accounting records;
- Official bookkeeper checks these records and vouches for them;
- Accounting records are business-critical; still ...
- Sometimes you need to be audited in your accounts.

Maps to three-party model:

- You are the Data Owner and want to keep accounting info private.
- Auditors (maybe the public in general), which need to be convinced that your accounting is correct, are the Service Provider.
- Official bookkeeper is the natural Trusted Source:
  - Already vouches for the data legally;
  - Could cryptographically *sign* data.

Same as before, but a notion of public verifiability (one-to-many authenticity) arises.

Related work / sources of inspiration (incomplete)

Privacy-preserving data processing:

[FKDL13] ZQL: A compiler for privacy-preserving data processing, Cédric Fournet, Markulf Kohlweiss, George Danezis, and Zhengqin Luo. USENIX Security, 2013.

Privacy-preserving smart metering, Alfredo Rial and George Danezis, Privacy in the Electronic Society, 2011.

Homomorphic authentication:

[CF13] Practical homomorphic MACs for arithmetic circuits, Dario Catalano and Dario Fiore. EUROCRYPT 2013.

Homomorphic signatures for polynomial functions, Dan Boneh and David Mandell Freeman. EUROCRYPT 2011.

(Nearly-)practical general zk-SNARK protocols:

[BCGTV13] SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge, Eli Ben-Sasson and Alessandro Chiesa and Daniel Genkin and Eran Tromer, Madars Virza, CRYPTO 2013

[BCTV14] Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture Eli Ben-Sasson and Alessandro Chiesa and Eran Tromer, Madars Virza USENIX Security 2014

[GGPR13] Quadratic span programs and succinct NIZKs without PCPs Rosario Gennaro, Craig Gentry, Bryan Parno, Mariana Raykova EUROCRYPT 2013

[PGHR13] Pinocchio: Nearly Practical Verifiable Computation Bryan Parno, Craig Gentry, Jon Howell, Mariana Raykova IEEE S&P 2013

What was lacking

Prior solutions lacked at least one of the following desirable features:

- Suitable for the 3-party model (i.e., allow proofs on secret authenticated data)
- General (i.e., not tied to a specific computation and apply to arbitrary computations)
- Scalable (i.e., degrades *graciously* for computations of increasing complexity and growing number of inputs)

Our goal is to achieve all three.

(Slightly more) Formal Model

Actors:

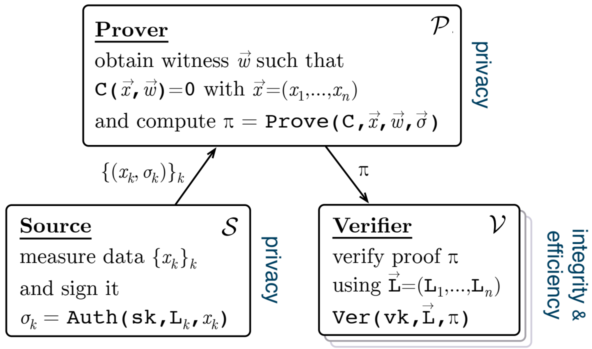
- A prover P (the data owner) proves computations $C(D)$ on data D to third parties V (the verifiers).
- The data D is generated and authenticated by trusted source S.

Security:

- Integrity $\rightarrow$ V should be convinced about the correctness of $C(D)$.
- Privacy $\rightarrow$ V should not learn any information about D beyond what is trivially revealed by $C(D)$.

Practicality:

- Data independence $\rightarrow$ source knows not what will be computed
- Prover scalability $\rightarrow$ should not cost much more than computing $C(D)$
- Verifier scalability $\rightarrow$ should cost much less than computing $C(D)$



Remark:
The link between authenticator and verifier is established via (public) labels a la homomorphic signatures and MACs.
Each piece of data is authenticated wrt to a unique label, whose semantics is application specific.

HASLab

zk-ADSNARK

Stands for *zero-knowledge Succinct Non-interactive ARguments of Knowledge on Authenticated Data*:

- Prove computations on secret authenticated data (or mix with public data).
- Zero knowledge $\rightarrow$ nothing extra about secret data (and witness) is revealed.
- Succinct $\rightarrow$ proof is short (constant size for given λ) and can be verified efficiently (linear time in the size of the data for given λ)
- Non-interactive $\rightarrow$ proof can be unilaterally constructed and sent.
- Argument of knowledge on authenticated data $\rightarrow$ successful verification on a set of labels implicitly defines an authenticated statement, and implies the extractability of a witness for it (aka adaptive soundness).

HASLab

zk-ADSNARK Syntax

- Global setup:
$$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$$
- Generate authentication keys:
$$(\mathsf{sk}, \mathsf{vk}, \mathsf{pap}) \leftarrow \mathsf{AuthKG}(\mathsf{pp})$$
- Authenticate one piece of data:
$$\sigma \leftarrow \mathsf{Auth}(\mathsf{sk}, L, x)$$
- Check authenticity of one piece of data:
$$\top/\bot \leftarrow \mathsf{AuthVer}(\mathsf{vk}, \sigma, L, x)$$
- Generate proving and verification keys
$$(\mathsf{EK}_C, \mathsf{VK}_C) \leftarrow \mathsf{Gen}(\mathsf{pap}, C)$$
- Generate proof
$$\pi \leftarrow \mathsf{Prove}(\mathsf{EK}_C, \vec{x}, \vec{w}, \vec{\sigma})$$
- Verify proof
$$\top/\bot \leftarrow \mathsf{Ver}(\mathsf{vk}, \mathsf{EK}_C, \vec{L}, \pi)$$

Implementation

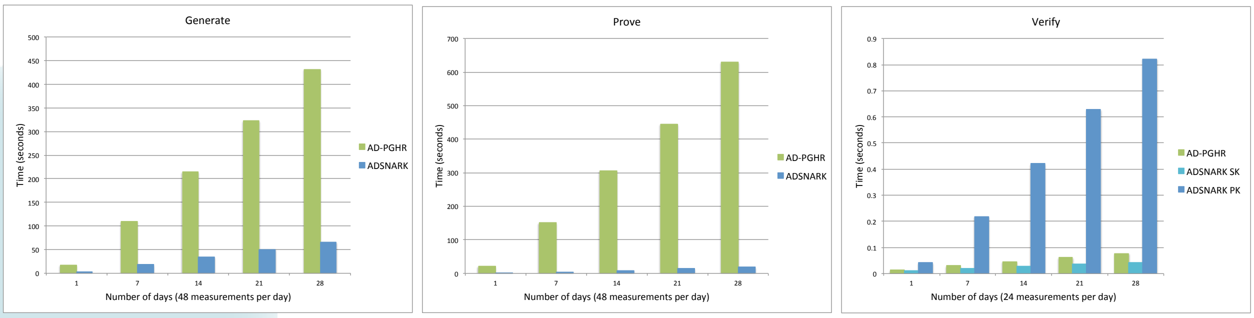
- Starting point: the libsnark zk-SNARK implementation. `https://github.com/scipr-lab/libsnark`
- Proof goals are expressed as systems of quadratic equations (constraint systems).
- Implemented simple translator from Pinocchio circuit format to libsnark constraint systems.
- Used previously existing circuit construction framework targeting Pinocchio to construct circuits for concrete applications.
- Extended libsnark with an implementation of our ADSNARK protocol.
- Additional components came from Supercop (`http://bench.cr.yp.to/supercop.html`):
  - Generic digital signature with extremely fast batch verification.
  - PRF built out of optimised AES implementation.

A concrete application

Typical smart metering scenario:

- Smart meter at your home produces a list of authenticated measurements.
- Price is given by a cumulative cost function, defined by a set of thresholds.
- For example:
  - The policy $[(0,2),(3,5),(7,8)]$
  - Establishes four consumption intervals and their corresponding prices,
    - $[0,3] \rightarrow 2$,
    - $(3,7] \rightarrow 5$,
    - $(7,\infty) \rightarrow 8$.
  - For consumption of 9, the price due is $3 \times 2 + 4 \times 5 + 2 \times 8 = 42$.
- Our hand-crafted arithmetic circuit for this computation takes $36 \times \#\text{measurements} \times \#\text{intervals} + 1$ multiplications. (Roughly the same number of constraints and variables)

Comparison for metering application



		Generation Time (seconds)		Proving Time (seconds)		Verification Time (seconds)		
Days	Mgates	AD-PGHR	ADSNARK	AD-PGHR	ADSNARK	AD-PGHR	ADSNARK SK	ADSNARK PK
1	8 641	17.929	3.262	21.760	0.622	0.013	0.013	0.042
7	60 481	110.164	18.296	151.146	4.463	0.030	0.020	0.219
14	120 961	214.457	34.507	306.705	9.078	0.047	0.028	0.421
21	181 441	213.647	50.770	444.592	14.314	0.062	0.037	0.628
28	241 921	431.341	65.539	629.003	18.426	0.077	0.043	0.823

		Proving Key Size (KBytes)		Verification Key Size (KBytes)		Proof size (Kbytes)		
Days	Mgates	AD-PGHR	ADSNARK	AD-PGHR	ADSNARK	AD-PGHR	ADSNARK SK	ADSNARK PK
1	8 641	17 463	2 500	1.9	1.9	0.3	0.4	6.4
7	60 481	124 274	17 641	10.9	10.9	0.3	0.4	42.4
14	120 961	248 547	35 282	21.3	21.4	0.3	0.4	84.4
21	181 441	364 661	52 923	31.8	31.8	0.3	0.4	126.4
28	241 921	497 094	70 563	42.2	42.3	0.3	0.4	168.4