

# Some privacy-enhancing technologies

Artem Kruglov, Skyori

# Disclaimer

- Third-party materials are used (links are provided at the end of the document)

# Motivation

- Modern cryptography is usually considered to have its beginning in the landmark papers of Diffie and Hellman, who introduced the concept of public key encryption, and of Rivest, Shamir and Adelman who suggested a concrete public key system. The fundamental theoretical studies along these lines originate in the late 1970's, and the results – the well-known cryptographic primitives of public key encryption, authentication and digital signature - have been widely applied in practice during the 1990's. However, theoretical cryptography provided additional, powerful (and perhaps less intuitive) tools.

# Motivation

- This review is a part of the preparation for practical use such technologies
- In our focus there are cryptographical methods and technologies that provides privacy using transformation data not division of data access
- Consider such technologies as primitivies for further business software development based on this primitives

# Content

- The purpose of the application of privacy-enhancing technologies
- Review some of privacy-enhancing technologies
- Conclusions

The purpose of the application of  
privacy-enhancing technologies

“...Privacy is necessary for an open society in the electronic age. Privacy is not secrecy. A private matter is something one doesn't want the whole world to know, but a secret matter is something one doesn't want anybody to know. Privacy is the power to selectively reveal oneself to the world. ....”

«A Cypherpunk's Manifesto» by Eric Hughes

# Privacy-enhancing technologies

- **Privacy-enhancing technologies (PETs)** are technologies that embody fundamental [data protection principles](#) by minimizing personal data use, maximizing data security, and empowering individuals.
- PETs have evolved since their first appearance in the 1980s. In intervals, review articles published the stats of privacy technology



# Examples of existing privacy enhancing technologies

- Communication anonymizers
- Shared bogus online accounts
- Obfuscation
- Access to personal data
- Enhanced privacy ID (EPID)
- Blinding
- Format-preserving encryption (FPE)
- Homomorphic encryption
- Zero-knowledge
- Secure multi-party computation

# Other PETs

- Limited disclosure technology,
- Anonymous credentials such as online car rental,
- Negotiation and enforcement of data handling conditions,
- Data transaction log
- ...

Some privacy-enhancing  
technologies

# Consider the following technologies

1. Secure multi-party computation
2. Zero-knowledge proof
3. Homomorphic encryption

|                        | Trusted Execution Environments   | Homomorphic Encryption  |               |                                     | Secure Multi-Party Computation   |                          | Personal Data Stores (b)   |
|------------------------|--|---|---------------|-------------------------------------|--|--------------------------|--|
| Type of privacy        | <ul style="list-style-type: none"> <li>Securely outsourcing to a server, or cloud, computations on sensitive data</li> </ul> | <ul style="list-style-type: none"> <li>Securely outsourcing specific operations on sensitive data</li> <li>Safely providing access to sensitive data</li> </ul>   |               |                                     | <ul style="list-style-type: none"> <li>Enabling joint analysis on sensitive data held by several organisations</li> </ul>  |                          | <ul style="list-style-type: none"> <li>Individual managing with whom and how they share data</li> <li>De-centralising services that rely on user data</li> </ul>                                 |
| Privacy risk addressed | <ul style="list-style-type: none"> <li>Revealing sensitive attributes present in a dataset</li> </ul>                        | <ul style="list-style-type: none"> <li>Revealing sensitive attributes present in a dataset</li> </ul>   |               |                                     | <ul style="list-style-type: none"> <li>Revealing sensitive attributes present in a dataset</li> </ul>  |                          | <ul style="list-style-type: none"> <li>Undesired sharing of sensitive information</li> </ul>   |
| Data protected         | <ul style="list-style-type: none"> <li>In storage</li> <li>During computing</li> </ul>                                       | <ul style="list-style-type: none"> <li>In storage</li> <li>During computing</li> </ul>  |               |                                     | <ul style="list-style-type: none"> <li>During computing</li> </ul>   |                          | <ul style="list-style-type: none"> <li>At point of collection</li> <li>During computing (locally)</li> </ul>   |
| Benefits               | <ul style="list-style-type: none"> <li>Commercial solutions widely available</li> <li>Zero loss of information</li> </ul>    | <ul style="list-style-type: none"> <li>Can allow zero loss of information</li> <li>FHE* can support the computation of any operation</li> </ul>   |               |                                     | <ul style="list-style-type: none"> <li>No need for a trusted third party - sensitive information is not revealed to anyone</li> <li>The parties obtain only the resulting analysis or model</li> </ul> |                          | <ul style="list-style-type: none"> <li>Gives full control to individuals</li> <li>Removes the risk of attacks on 'honeypots' of centralised data</li> <li>Analysis can be run locally</li> </ul> |
| Current limitations    | <ul style="list-style-type: none"> <li>Many side-channel attacks possible</li> </ul>   | <ul style="list-style-type: none"> <li>FHE currently inefficient, but SHE* and PHE* are usable</li> <li>Highly computationally intensive; bandwidth and latency issue</li> <li>Running time</li> <li>PHE and SHE support the computation of limited functions</li> <li>Standardisation in progress</li> </ul> |               |                                     | <ul style="list-style-type: none"> <li>Highly compute and communication intensive</li> </ul>   |                          | <ul style="list-style-type: none"> <li>Impracticality of individual controlling data sharing with many parties</li> </ul>  |
| Readiness level        | Product  | PHE:<br>Product   | SHE:<br>Pilot | FHE:<br>Research – proof of concept | PSI*, PIR*:<br>Product   | Proof of concept – pilot | Product  |
| Qualification criteria |  | <ul style="list-style-type: none"> <li>Specialist skills</li> <li>Custom protocols</li> <li>Computing resources</li> </ul>  |               |                                     | <ul style="list-style-type: none"> <li>Specialist skills</li> <li>Custom protocols</li> <li>Computing resources</li> </ul>   |                          |  |

# Secure multi-party computation

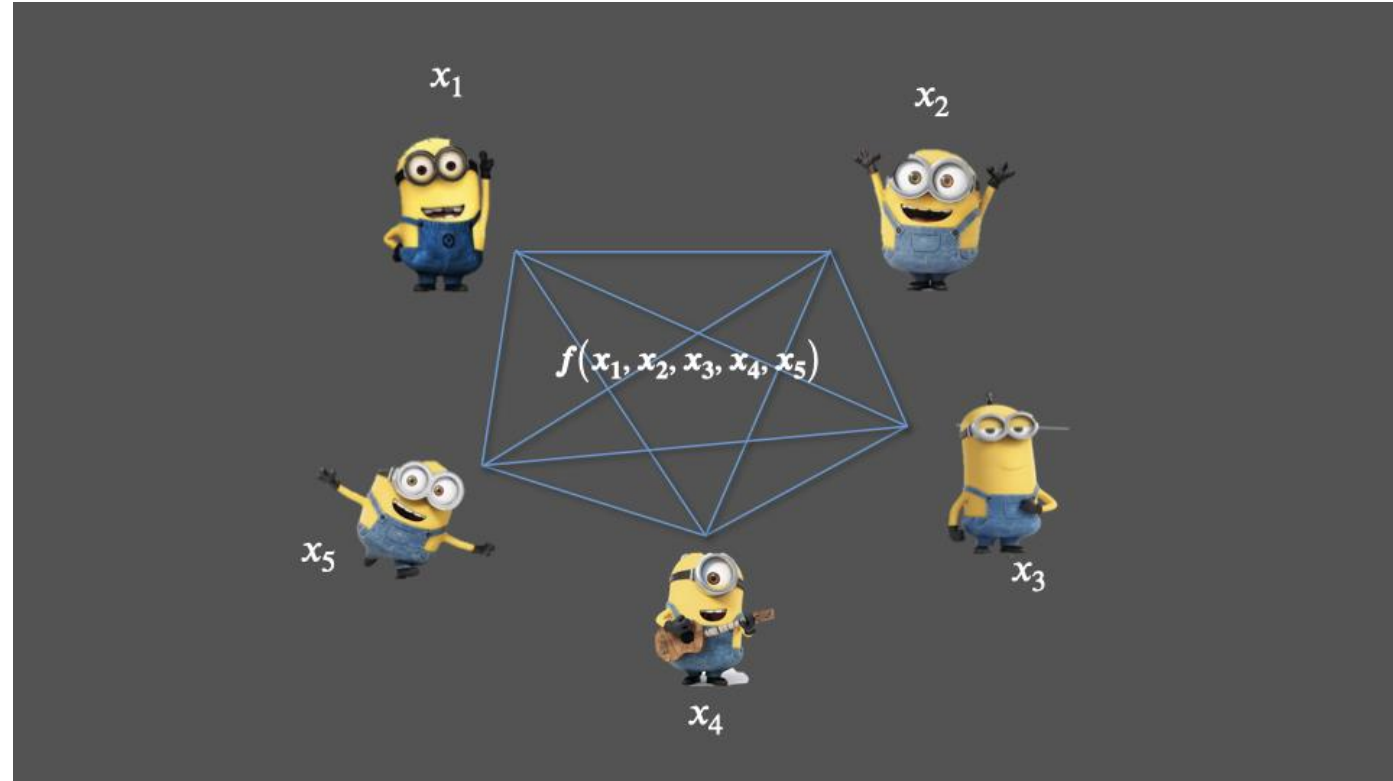
Secure Function Evaluation (SFE)

Secure Multiparty Computation (SMC)

Multi-Party Computation (MPC)

# Main idea

parties jointly compute a function over their inputs while keeping those inputs private



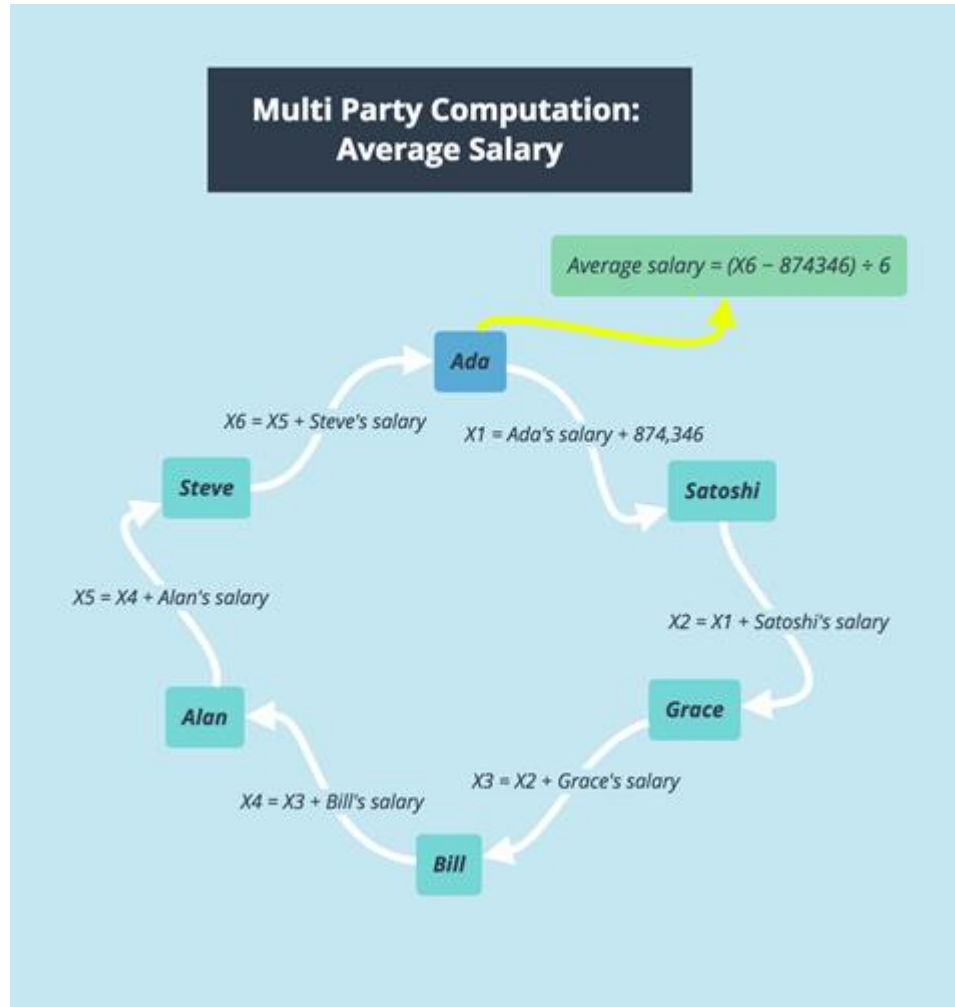
# Riddle (on russian)

А Б В Г Д Е Ё Ж З И Й К Л М Н О П  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я  
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33



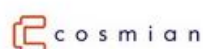
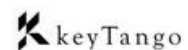
# Main idea



|                          |                 |                      |
|--------------------------|-----------------|----------------------|
| Secret X: 42             | $X = X_1 + X_2$ | $X_1 = 50, X_2 = -8$ |
| Secret Y: 23             | $Y = Y_1 + Y_2$ | $Y_1 = 20, Y_2 = 3$  |
| Compute $X + Y = ?$ (65) |                 |                      |
| Computer 1               | Computer 2      |                      |
| $X_1 + Y_1$              | $X_2 + Y_2$     |                      |
| $50 + 20 = 70$           | $-8 + 3 = -5$   | $70 - 5 = 65$        |

# Application

- threshold cryptography and key management
- private multi-party machine learning and data analysis (<https://www.soda-project.eu/> )
- private information retrieval (from DBs)
- private DNA comparisons,
- privacy-preserving auctions,
- privacy-preserving appointment scheduling
- privacy-preserving .... anythings 😊



# How to use this technology

- <https://github.com/topics/secure-computation> 40
- <https://github.com/topics/multiparty-computation> 36
- <https://github.com/topics/multi-party-computation> 23
- <https://github.com/topics/multiparty> 20
  
- private AI technologies <https://www.openmined.org/>

Zero-knowledge proof


# Main idea

A family of protocols that verifies the truth of a certain statement without disclosing it.

Example:

- **proof that you ran a computer program on some input and you got some output.** If you run a program  $P$  with some input  $x$  and get some output  $y = P(x)$ , you can generate a ZK-SNARK, which you can give to someone else to convince someone else that  $y$  actually is the result of running  $P(x)$ .
- **The proof can be verified much more quickly than running  $P$  directly,** so it's a powerful tool for trustlessly outsourcing computation.

# zkSNARK definition

Generator (C circuit,  $\lambda$  is ):

$(pk, vk) = G(\lambda, C)$

Prover (x pub inp, w sec inp):

$\pi = P(pk, x, w)$

Verifier:

$V(vk, x, \pi) == (\exists w \text{ s.t. } C(x, w))$

```
function C(x, w) {  
    return ( sha256(w) == x );  
}
```

## Range (Age) proof example

$$F(X, Y) = Z$$

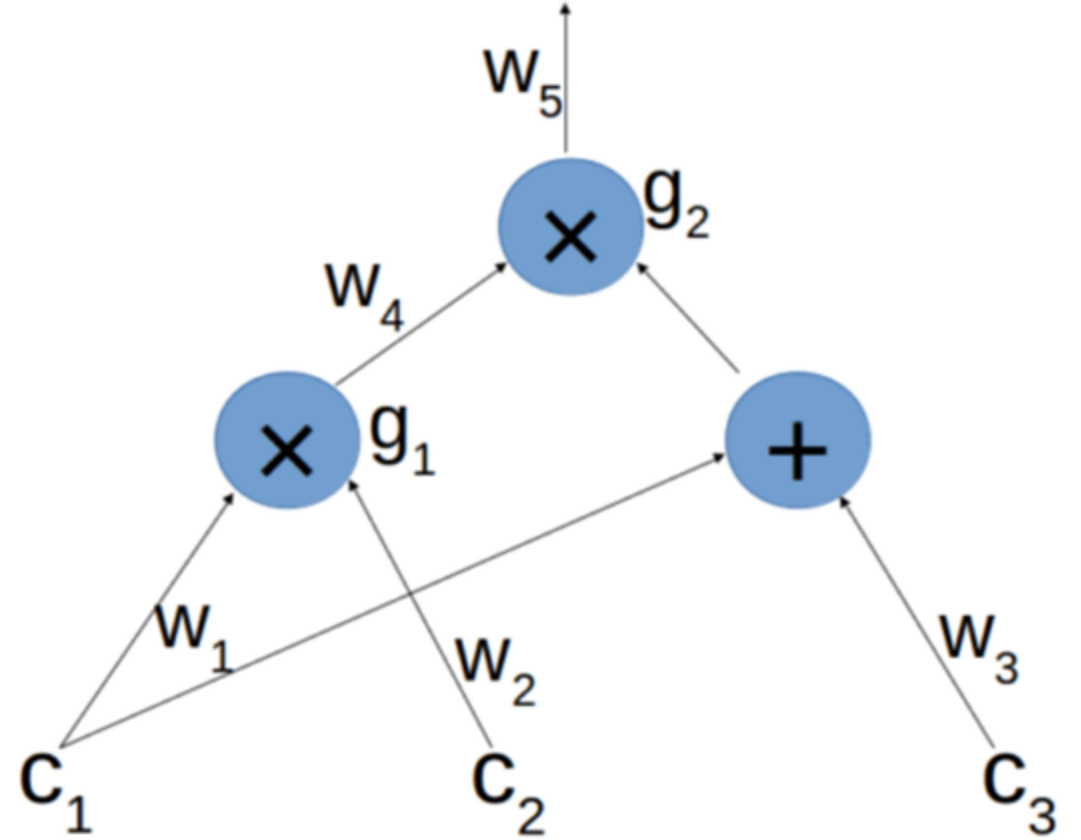


**Proof**

Here is **X** and **Z**, I know of an **Y**  
such that  $F(X, Y) = Z$



- Computation
- → Arithmetic Circuit
- → R1CS (Rank 1 Constraint System)
- → QAP (Quadratic Arithmetic Program)
- → zk-SNARK



# Trusted setup

- Procedure of generation trusted parameters
- After its finished, rules and parameters of this procedure can not be verified, you can only trust it. You can verify it if only you are participant of this procedure.

# ZKP family

|                               | Proof Size | Prover Time | Verification Time |
|-------------------------------|------------|-------------|-------------------|
| SNARKs<br>(has trusted setup) | 288 bytes  | 2.3s        | 10ms              |
| STARKs                        | 45KB-200KB | 1.6s        | 16ms              |
| Bulletproofs                  | ~1.3KB     | 30s         | 1100ms            |

# Application

- Cryptocurrency private transactions
- Range proofs
- Membership/non-membership proofs
- Correct behavior proofs
- ...

# How to use this technology

- <https://hackmd.io/@zkteam/gnark>
- <https://github.com/ConsenSys/gnark>
- <https://zinc.matterlabs.dev/>
  
- <https://github.com/topics/zero-knowledge> 122
- <https://github.com/topics/zero-knowledge-proofs> 78
- <https://github.com/topics/zk-snarks> 66
- <https://github.com/topics/zkp> 19

Homomorphic encryption

# Homomorphic encryption

- Homomorphic encryption is a form of [encryption](#) with an additional evaluation capability for computing over encrypted data without access to the [secret key](#)
- [Craig Gentry](#), using [lattice-based cryptography](#), described in 2009 the first plausible construction for a fully homomorphic encryption scheme
- *Partially homomorphic encryption* encompasses schemes that support the evaluation of circuits consisting of only one type of gate, e.g., addition or multiplication
- <https://github.com/topics/homomorphic-encryption> 135

# Compare with some PETs

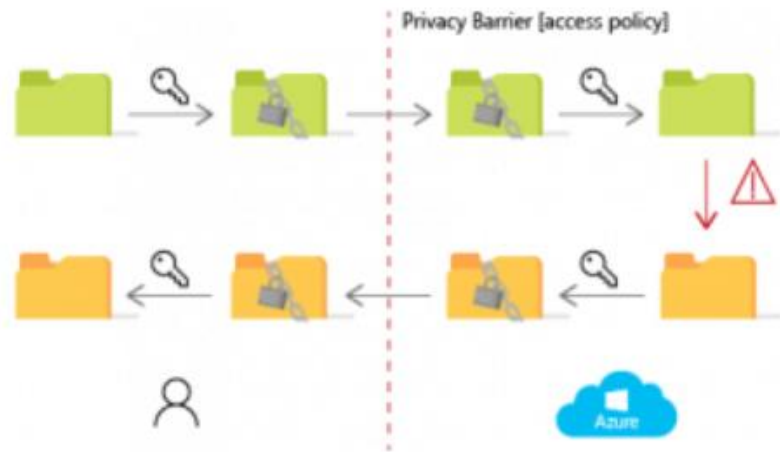
- Homomorphic encryption is indeed powerful, but it's a quite different tool than ZK-SNARKs:
  - In general, doing things with ZK-SNARKs requires all of the private information to be held by one party (the prover) at the point in time.
  - **Homomorphic encryption**, obfuscation, MPC, and the rest are valuable because they enable computations where **different inputs are held by different people** and neither party learns the private information of the others.
  - In the case of homomorphic encryption, one party retains the privacy of their data, and the other party retains the privacy of what program they ran on the data. You *could* also use homomorphic encryption for computation outsourcing for scalability, but in practice **the overhead of homomorphic encryption is too high for that to be practical at present.**



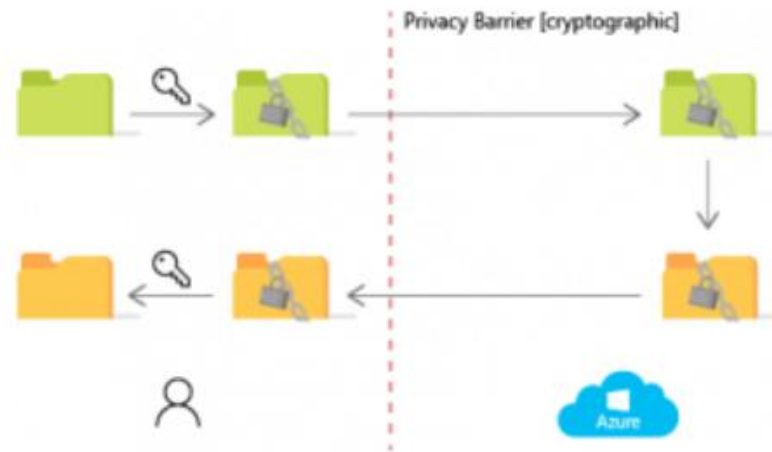
# Application

- Secure computing

Traditional Cloud Storage and Computation



Microsoft SEAL Cloud Storage and Computation



# Application

- Privacy-preserving machine learning using homomorphic encryption
- Private AI technologies <https://www.openmined.org/>
- Secure genome analysis using Homomorphic Encryption - <http://www.humangenomeprivacy.org>
- COVID Alert <https://coronatrace.org/>
- Homomorphic encryption as a service <https://morfix.io>

# Participants in standardization workshops from the industry

- Microsoft
- Samsung SDS
- Intel
- [Duality Technologies](#)
- IBM
- [Inpher](#)
- Google
- SAP
- Intuit
- General Dynamics
- Mastercard
- CryptoExperts
- Algorand Foundation
- Mercedes Benz
- Alibaba Group
- LinkedIn
- [IXUP](#)
- Intertrust

# How to use this technology

- [HELib](#): An early and widely used library from IBM that supports the BGV scheme and bootstrapping.
- [Microsoft SEAL](#): A widely used open source library from Microsoft that supports the BFV and the CKKS schemes.
- [PALISADE](#): A widely-used open source library from a consortium of DARPA-funded defense contractors that provides lattice cryptography building blocks and supports leading homomorphic encryption schemes.
- [FHEW](#) / [TFHE](#): (Torus-FHE) GSW-based libraries with fast bootstrapped operations. TFHE is designed from FHEW, which is no longer actively being developed.
- [HeaAn](#): This library implements the CKKS scheme with native support for fixed point approximate arithmetic.
- [Λ o λ](#) (pronounced “L O L”): This is a Haskell library for ring-based lattice cryptography that supports FHE.
- [NFLlib](#): This library is an outgrowth of the European HEAT project to explore high-performance homomorphic encryption using low-level processor primitives.
  - [HEAT](#): This library focuses on an API that bridges FV-NFLib and HeLIB.
  - [HEAT](#): A HW accelerator implementation for FV-NFLlib.
- [cuHE](#): This library explores the use of GPGPUs to accelerate homomorphic encryption.
- [Lattigo](#): This is a lattice-based cryptographic library written in Go.
- <https://homomorphicencryption.org/>

Conclusions

# Conclusions

- Privacy-enhancing technologies are difficult for using in practice and are needed mathematical background of software developers
- There are many example on difference programming languages
- There are some projects that creating frameworks and platforms for simplify using such privacy technologies aimed to be an enterprise-level software development tools

Links

# Links

- Homomorphic Encryption <https://homomorphicencryption.org/>
- the iDASH project on Secure Genome Analysis  
<http://www.humangenomeprivacy.org>
- <https://www.microsoft.com/en-us/research/project/microsoft-seal/>
- Awesome Homomorphic Encryption  
<https://github.com/jonaschn/awesome-he>
- OpenMined, private AI technologies. <https://www.openmined.org/>



# Links

- the MPC alliance <https://www.mpcalliance.org/>
- D. Malkhi, N. Nisan, B. Pinkas and Y. Sella. Fairplay — a secure two-party computation system. In Proc. of 13th USENIX Security Symposium, 2004.
- A Cypherpunk's Manifesto by [Eric Hughes](#)
- [https://en.wikipedia.org/wiki/Privacy-enhancing\\_technologies](https://en.wikipedia.org/wiki/Privacy-enhancing_technologies)
- [https://en.wikipedia.org/wiki/Homomorphic\\_encryption](https://en.wikipedia.org/wiki/Homomorphic_encryption)
- <https://royalsociety.org/-/media/policy/projects/privacy-enhancing-technologies/privacy-enhancing-technologies-report.pdf>