BetterTimes

Privacy-assured Outsourced Multiplications for Additively Homomorphic Encryption on Finite Fields

Per Hallgren [1]
Martín Ochoa [2,3]
Andrei Sabelfeld [1]

1. Chalmers University of Technology
2. Technische Universität München
3. Singapore University of Technology and Design

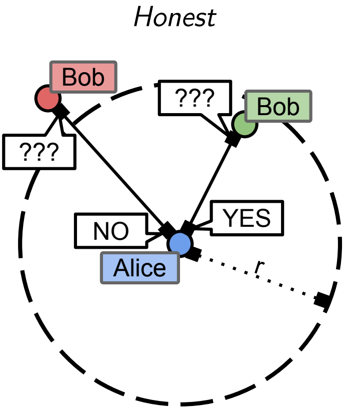
November 20, 2015

The problem of honest-but-curious adversaries

- Using a too weak attacker model can have serious consequences

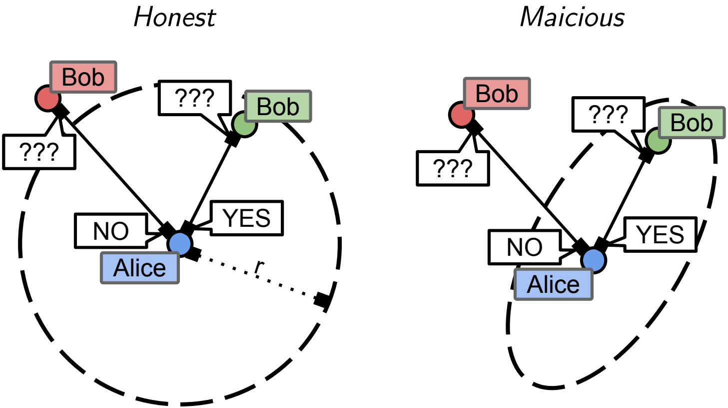
The problem of honest-but-curious adversaries

- Using a too weak attacker model can have serious consequences



The problem of honest-but-curious adversaries

- Using a too weak attacker model can have serious consequences



Arithmetic formulas

- Many privacy-preserving solutions use arithmetic formulas
  - Privacy-preserving face recognition
  - Privacy-preserving location proximity
  - Privacy-preserving auctioning and bartering systems
  - Privacy-preserving voting

Arithmetic formulas

- Many privacy-preserving solutions use arithmetic formulas
  - Privacy-preserving face recognition
  - Privacy-preserving location proximity
  - Privacy-preserving auctioning and bartering systems
  - Privacy-preserving voting
- Common assumption is honest-but-curious

Arithmetic formulas

- Many privacy-preserving solutions use arithmetic formulas
  - Privacy-preserving face recognition
  - Privacy-preserving location proximity
  - Privacy-preserving auctioning and bartering systems
  - Privacy-preserving voting
- Common assumption is honest-but-curious
- Many current solutions suffer
  - Face recognition: Sadeghi et al. 2009, Erkin et al. 2009
  - Location proximity: Zhong et al. 2007, Sedenka and Gasti 2014, Hallgren et al. 2015

General Problem

- Privacy-assurances when computing arithmetic formulas in the malicious model

General Problem

- Privacy-assurances when computing arithmetic formulas in the malicious model
    - ***Privacy*** against malicious adversaries
        - Can lie about their inputs
        - Can potentially give false outputs to the other party
        - Can ***not*** learn anything about the other parties outputs

General Problem

- Privacy-assurances when computing arithmetic formulas in the malicious model
    - ***Privacy*** against malicious adversaries
        - Can lie about their inputs
        - Can potentially give false outputs to the other party
        - Can ***not*** learn anything about the other parties outputs
    - Two-party setting
        - Two principals *Alice* (A) and *Bob* (B)
        - Alice is the initiating party, and Alice receives the output

General Problem

- Privacy-assurances when computing arithmetic formulas in the malicious model
  - ***Privacy*** against malicious adversaries
    - Can lie about their inputs
    - Can potentially give false outputs to the other party
    - Can ***not*** learn anything about the other parties outputs
  - Two-party setting
    - Two principals *Alice* (A) and *Bob* (B)
    - Alice is the initiating party, and Alice receives the output
  - Goal
    - Bob learns nothing
    - Alice learns *at most* the intended output

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$
- plaintext space $\mathcal{M}$ isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$
- plaintext space $\mathcal{M}$ isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$
- Key properties we will use
  - *Addition*: $[\![m_1 + m_2]\!] = [\![m_1]\!] \oplus [\![m_2]\!]$

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$
- plaintext space $\mathcal{M}$ isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$
- Key properties we will use
  - *Addition*: $[\![m_1 + m_2]\!] = [\![m_1]\!] \oplus [\![m_2]\!]$
  - *Subtraction*: $[\![m_1 - m_2]\!] = [\![m_1]\!] \ominus [\![m_2]\!]$

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$
- plaintext space $\mathcal{M}$ isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$
- Key properties we will use
  - *Addition*: $[\![m_1 + m_2]\!] = [\![m_1]\!] \oplus [\![m_2]\!]$
  - *Subtraction*: $[\![m_1 - m_2]\!] = [\![m_1]\!] \ominus [\![m_2]\!]$
  - *Multiplication*: $[\![m_1 \cdot m_2]\!] = [\![m_1]\!] \odot m_2$

The solution is based upon Homomorphic Encryption

- k, K private and public key.
  - Private key held by Alice
  - Public key globally known
- The encryption of a plaintext p using K is denoted as $[\![p]\!]$
- plaintext space $\mathcal{M}$ isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$
- Key properties we will use
  - *Addition*: $[\![m_1 + m_2]\!] = [\![m_1]\!] \oplus [\![m_2]\!]$
  - *Subtraction*: $[\![m_1 - m_2]\!] = [\![m_1]\!] \ominus [\![m_2]\!]$
  - *Multiplication*: $[\![m_1 \cdot m_2]\!] = [\![m_1]\!] \odot m_2$
  - *Blinding*: given $\mathcal{M}^{\mathcal{U}}$ uniformly random distribution in $\mathcal{M} \setminus \{0\}$
    - $[\![m]\!] \oplus [\![b]\!] = [\![r]\!]$, with $b, r \in \mathcal{M}^{\mathcal{U}}$
    - $[\![m]\!] \odot [\![b]\!] = [\![r]\!]$, with $b, r \in \mathcal{M}^{\mathcal{U}}$

Extension to additively homomorphic encryption

- Since an additively homomorphic encryption system has

Extension to additively homomorphic encryption

- Since an additively homomorphic encryption system has
    - Addition
    - Subtraction

Extension to additively homomorphic encryption

- Since an additively homomorphic encryption system has
  - Addition
  - Subtraction
  - Multiplication *with one known plaintext*

Extension to additively homomorphic encryption

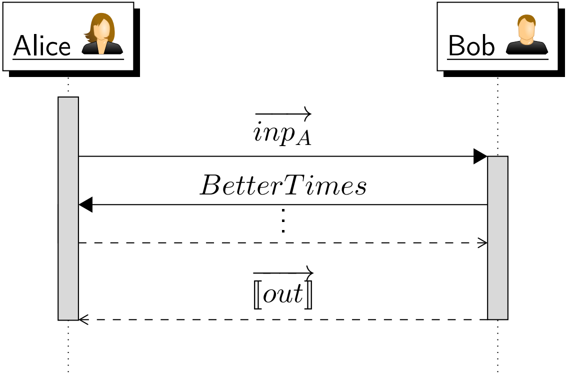
- Since an additively homomorphic encryption system has
  - Addition
  - Subtraction
  - Multiplication *with one known plaintext*
- The only thing we need to add is $[\![m_1]\!] \odot [\![m_2]\!] = [\![m_1 \cdot m_2]\!]$

Extension to additively homomorphic encryption

- Since an additively homomorphic encryption system has
  - Addition
  - Subtraction
  - Multiplication *with one known plaintext*
- The only thing we need to add is $[\![m_1]\!] \odot [\![m_2]\!] = [\![m_1 \cdot m_2]\!]$
- We solve this using *outsourcing* these multiplications through a novel protocol called *BetterTimes*.

Communication Overview

- In our setting, protocols follow the form
  - Alice initiates the protocol
  - Bob sees only encrypted data (he can't decrypt)
  - Possibly there are more round trips to finish the computation
  - Bob responds with the final result



BetterTimes

The protocol is outlined as follows:

1. BetterTimes is run when Bob wants to multiply $[\![x]\!]$ and $[\![y]\!]$

BetterTimes

The protocol is outlined as follows:

1. BetterTimes is run when Bob wants to multiply $[\![x]\!]$ and $[\![y]\!]$
2. He sends the blinded $[\![x']\!]$, $[\![y']\!]$, *challenge* $[\![c]\!]$ to Alice

BetterTimes

The protocol is outlined as follows:

1. BetterTimes is run when Bob wants to multiply $[\![x]\!]$ and $[\![y]\!]$
2. He sends the blinded $[\![x']\!]$, $[\![y']\!]$, *challenge* $[\![c]\!]$ to Alice
3. Alice replies with $[\![z']\!](= [\![x' \cdot y']\!])$ and *assurance* $[\![a']\!]$

BetterTimes

The protocol is outlined as follows:

1. BetterTimes is run when Bob wants to multiply $[\![x]\!]$ and $[\![y]\!]$
2. He sends the blinded $[\![x']\!]$, $[\![y']\!]$, *challenge* $[\![c]\!]$ to Alice
3. Alice replies with $[\![z']\!](= [\![x' \cdot y']\!])$ and *assurance* $[\![a']\!]$
4. Bob removes the blinding from $[\![z']\!]$ to arrive at $[\![z]\!]$

BetterTimes

The protocol is outlined as follows:

1. BetterTimes is run when Bob wants to multiply $[\![x]\!]$ and $[\![y]\!]$
2. He sends the blinded $[\![x']\!]$, $[\![y']\!]$, *challenge* $[\![c]\!]$ to Alice
3. Alice replies with $[\![z']\!](= [\![x' \cdot y']\!])$ and *assurance* $[\![a']\!]$
4. Bob removes the blinding from $[\![z']\!]$ to arrive at $[\![z]\!]$
5. Bob computes $[\![a]\!]$ using all of $[\![x']\!]$, $[\![y']\!]$, $[\![z']\!]$ and $[\![a']\!]$

BetterTimes communication

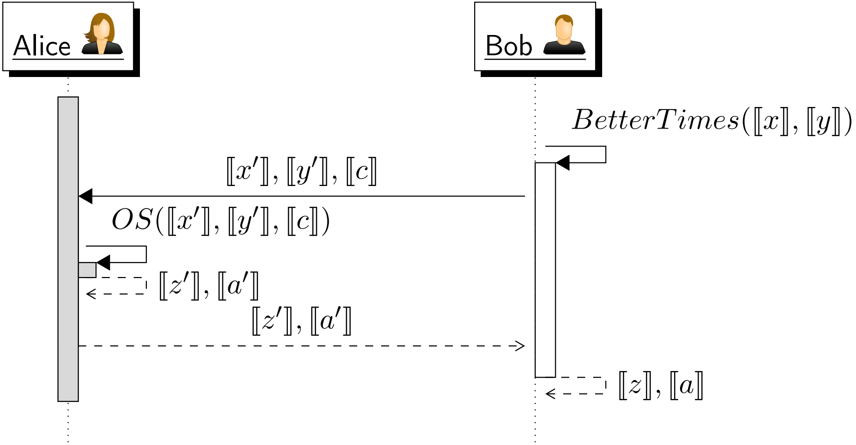


Figure: Visualization of the attested multiplication protocol

Using BetterTimes in a formula

- Bettertimes assures that $[\![a]\!]$ is zero if and only if $[\![z]\!] = [\![x \cdot y]\!]$, and a uniformly random value otherwise.

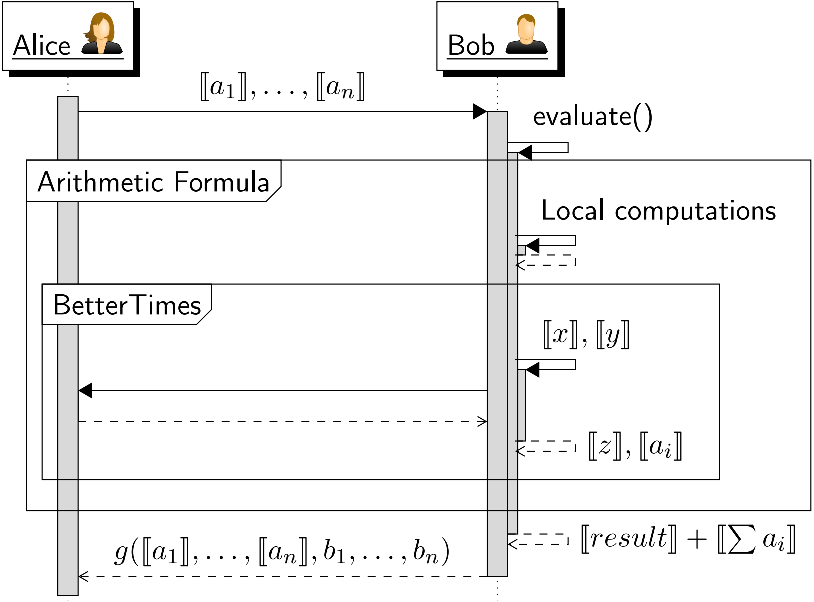
Using BetterTimes in a formula

- Bettertimes assures that $[\![a]\!]$ is zero if and only if $[\![z]\!] = [\![x \cdot y]\!]$, and a uniformly random value otherwise.
- When Bob has computed the final result $[\![result]\!]$, he sends $[\![result]\!] + \sum a_i$ to Alice, where a_i is the assurance value corresponding to each outsourced multiplication.

Using BetterTimes in a formula

- Bettertimes assures that $[\![a]\!]$ is zero if and only if $[\![z]\!] = [\![x \cdot y]\!]$, and a uniformly random value otherwise.
- When Bob has computed the final result $[\![result]\!]$, he sends $[\![result]\!] + \sum a_i$ to Alice, where a_i is the assurance value corresponding to each outsourced multiplication.
- Alice receives the correct output if and only if she computed all outsourced multiplications honestly, and a uniformly random value otherwise

Private Evaluation of Arithmetic Formula



Proof Outline for Privacy of Arbitrary Formula

Our Privacy definition follows the standard framework for secure multi-part computation (Lindell and Pinkas 2008)

Theorem

For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ *represented by a recursive instruction* $\iota \in$ ***Ins****, for every adversary* $\mathcal{A}$ *against the protocol* π *resulting from* $evaluate(\iota)$*, there exist a simulator* $\mathcal{S}$ *such that:*

$$\{\mathsf{IDEAL}_{g,\mathcal{S}(s)}(\overrightarrow{x}, \overrightarrow{y})\} \overset{c}{\equiv} \{\mathsf{REAL}_{\pi,\mathcal{A}(s)}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\overset{c}{\equiv}$ *denotes computational indistinguishability of distributions.*

Proof Outline for Privacy of Arbitrary Formula

Our Privacy definition follows the standard framework for secure multi-part computation (Lindell and Pinkas 2008)

Theorem

*For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction $\iota \in$ **Ins**, for every adversary $\mathcal{A}$ against the protocol π resulting from $evaluate(\iota)$, there exist a simulator $\mathcal{S}$ such that:*

$$\{\mathsf{IDEAL}_{g,\mathcal{S}(s)}(\overrightarrow{x}, \overrightarrow{y})\} \stackrel{c}{\equiv} \{\mathsf{REAL}_{\pi,\mathcal{A}(s)}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability of distributions.

- The full proof is given in the paper

Proof Outline for Privacy of Arbitrary Formula

Our Privacy definition follows the standard framework for secure multi-part computation (Lindell and Pinkas 2008)

Theorem

*For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction $\iota \in$ **Ins**, for every adversary $\mathcal{A}$ against the protocol π resulting from $evaluate(\iota)$, there exist a simulator $\mathcal{S}$ such that:*

$$\{\mathsf{IDEAL}_{g,\mathcal{S}(s)}(\overrightarrow{x}, \overrightarrow{y})\} \stackrel{c}{\equiv} \{\mathsf{REAL}_{\pi,\mathcal{A}(s)}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability of distributions.

- The full proof is given in the paper
- The theorem implicates that **any protocol** evaluating arithmetic formulas as defined in the paper can be evaluated in the presence of a malicious adversary while preserving privacy

Benchmarks

- Performed benchmarks on prototype implementation in python
- Comparing to outsourced multiplications secure only against honest adversaries

Table: Times (in milliseconds) for outsourced multiplication

Plaintext space	Time (in milliseconds)					
	1024 bits			2048 bits		
	This approach	Naive approach	Extra work	This approach	Naive approach	Extra work
2^2	6.286	4.016	56.52%	29.686	19.458	52.56%
2^8	6.400	4.017	59.32%	30.052	19.484	54.24%
2^{16}	6.432	4.148	55.06%	30.188	19.574	54.22%
2^{24}	6.538	4.100	59.46%	30.578	19.801	54.43%

- Benchmarks show that our more secure approach costs about 53-60% extra work for a multiplication

Protocols that can be secured with BetterTimes

- Several existing works can use the proposed approach to increase protection against malicious attackers
  - Privacy-preserving face recognition: Sadeghi et al. 2009, Erkin et al. 2009
  - Privacy-preserving location proximity: Zhong et al. 2007, Sedenka and Gasti 2014, Hallgren et al. 2015

Conclusions

- Presented BetterTimes
- Using BetterTimes one can compute any arithmetic formula in the presence of a malicious Alice
- The overhead, compared to protection against honest adversaries, is about 55%
  - Of each multiplication, not of the formula as a whole
  - Usually the number of multiplications is minimized, as additions are cheap with additively homomorphic encryption

Thank you for your attention!

Questions?